FORECAST USA'S ELECTRICAL ENERGY GENERATION

S2-22_DSECCZG628T: Dissertation

by

Kusal Chand Prakash

2021c104050

Dissertation work carried out at

LTIMindtree, Merlin Infinite, Block-DN,
Sector-V Salt Lake Electronics Complex,
Kolkata, West Bengal 700091

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
Pilani (Rajasthan) India

October, 2023

FORECAST USA'S ELECTRICAL ENERGY GENERATION

S2-22_DSECCZG628T: Dissertation

Submitted in partial fulfilment of the requirements of

MTech. Data Science & Engineering

by

Kusal Chand Prakash
2021c104050

Under the supervision of

Mr. Anirban Pramanik, Senior Account Director

Project Work carried out at

LTIMindtree, Merlin Infinite, Block-DN,
Sector-V Salt Lake Electronics Complex,
Kolkata, West Bengal 700091

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE
Pilani (Rajasthan) India

October, 2023

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

CERTIFICATE

This is to certify that the Project Work entitled **Forecast USA's electrical energy generation** and submitted by **Kusal Chand Prakash** ID No. **2021c104050** for the partial fulfillment of the requirements of **S2-22_DSECCZG628T: Dissertation**, embodies the work done by him under my supervision.

A. Pramanik

_____

Signature of the Supervisor

Mr. Anirban Pramanik

Senior Account Director
LTIMindtree

Date: 09 Oct 2023

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

S2-22_DSECCZG628T: Dissertation

Project Work Title : Forecast USA's future electrical energy generation

Name of Supervisor : Mr. Anirban Pramanik

Name of Student : Kusal Chand Prakash

BITS ID No. of Student : 2021C104050

Abstract *(around 250 words)*:
Electrical energy is the backbone of a country, more in the case of USA as it is a superpower. It is indispensable for supporting and powering USA's critical infrastructure, industry, services, and daily activities that are essential for the functioning and the development of USA.

The law of 'Energy conservation' states that 'Energy can neither be created nor destroyed, it can only be transferred or transformed from one form to another.' Energy cannot be 'stored.'

If USA produces more electrical energy than it needs, it can face challenges such as oversupply, which can lead to wasted resources, increased costs for storage and potential strain on the electrical grid.

If USA produces less electrical energy than it needs, it can experience power shortages, leading to disruptions in daily activities, economic setbacks, and potential impacts on the quality of life for its citizens. It can also make USA dependent on energy imports hence making it vulnerable to supply disruptions, price fluctuations and geopolitical factors, potentially affecting its energy security.

Hence, the challenge for USA to produce the optimal amount of energy. To achieve this USA needs to predict with a fair degree of accuracy its annual energy needs and prepare for the same (in the long and short term).

In this dissertation we are going to use USA's historical data of electrical energy generation vs consumption, population, weather elements, per-capita-income etc. to predict its electrical energy needs for the next month or next year.
With the right data this model could be tweaked to predict the electrical energy requirement for any other country.

Key Words (in Alphabetical order): ANN, ARIMA, Auto Regression, LSTM, prediction, RNN, SARIMAX, stationary data, XGBoost.

Acknowledgements

I would like to express my heartfelt gratitude to the following stakeholders.

Cognizant who gave me the opportunity to purse this course.

LTIMindtree who gave me the opportunity to continue and complete this course.

My supervisor, Mr. Anirban Pramanik for his invaluable guidance, unwavering support, and expertise throughout the completion of this dissertation. During the commencement of my dissertation, he asked me to come up with some topics on my own. He appreciated the topic which I had finally chosen and provided treasured insights. He taught me 'how to catch a fish.'

I appreciate the prized resources and research papers shared by Dr. Sujatha Mam, BITS Pilani. Her assistance was akin to the guiding beacon of a lighthouse, steering my ship.

I am also thankful to my family and friends for their encouragement and understanding during this journey.

This work would not have been possible without everyone's collective support. Thank you all for being an integral part of my academic endeavor.

Table of Contents

1. Acronyms

Serial number	Acronym	Comments
1.	ANFIS	Adaptive Neuro Fuzzy Inference System
2.	ARIMA	Auto Regressive Integrated Moving Average
3.	ARMA	Auto Regressive Moving Average
4.	BPTT	Back Propagation Through Time
5.	CDD	Cooling Degree Days
6.	DE	Differential Evolutionary algorithm
7.	DL	Deep Learning
8.	HDD	Heating Degree Days
9.	kWH	kilo Watt-Hour
10.	LSTM	Long Short-Term Memory
11.	MAM	Moving Average Model
12.	MAPE	Mean Absolute Percentage Error
13.	m/c	machine
14.	mkWH	million kilo Watt Hours
15.	ML	Machine Learning
16.	MSE	Mean Squared Errors
17.	NLP	Natural Language Processing
18.	NN	Neural Network
19.	RDPI	Real Disposable Personnel Income
20.	RF	Random Forest
21.	RNN	Recurrent Neural Network
22.	SVR	Support Vector Regression
23.	tsd	time-series-data
24.	VAR	Vector Auto Regression

2. List of images

3. List of important tables

4. Introduction

We got an insight from the 'Abstract' above how producing the right amount of electrical energy is indispensable for USA. Even more important is its prediction. From the literature review below, it is evident that predicting electrical energy generation for an entire country has fairly been done and that too using multivariate analysis. We will attempt to do so in this dissertation. Below is the summary of the chapters present in this dissertation.

Chapter 1: Data – data sources, data ingestion, data pre-processing etc.
Chapter 2: Exploratory data analysis (EDA) – Data exploration and visualization, feature extraction etc.
Chapter 3: Performance metrics used in the evaluation of the time series forecasting models
Chapter 4: Auto regression ARIMA, SARIMA, SARIMAX
Chapter 5: XGBoost
Chapter 6: LSTM
Chapter 7: Conclusion and Future work

5. Literature review

Research paper 1	Prediction of energy demands using Neural Network with model identification. Link
Algorithm / Technique / Model / Method used	Global optimization method called the Modal trimming method, Quasi-Newton method.
Data set and attributes set used	23-week days in July 1996, sampling time interval = 1 hour. Electrical energy, temperature, humidity used as supplementary conditions.
Advantages	Prediction using global optimization method is more accurate than the local optimization method.
Suggestions / Future enhancement/ Research gap	Only 2 dependent features used. The consideration of supplemental conditions can improve the accuracy of the identification of the values of model parameters for the neural network model, but it may deteriorate the accuracy of the prediction. Hence effect of other features like population, weather elements (like HDD, CDD), RDPI on energy generation cannot be determined. Does not provide an annual prediction of electrical energy.

Research paper 2	Time series forecasting for building energy consumption using weighted Support Vector Regression with differential evolution optimization technique. Link
Algorithm / Technique / Model / Method used	Weighted-hybrid SVR model. Parameter selection for the SVR models and optimization of the weights is done by the Differential Evolutionary algorithm.
Data set and attributes set used	Electrical energy data (kWH) requirement of a building in Singapore for a year. Electrical energy.
Advantages	SVR models with optimization algorithm can achieve overall high forecasting accuracy due to its effectiveness in modeling complex and nonlinear data series. The proposed model was applied to two datasets for the same building. The two datasets are that of daily energy consumption for a period of one year and half hourly energy consumption for a period of ten days. It is observed that the proposed model can forecast the energy consumption for both the datasets with good accuracy, without changing the model structure. This is achieved by using the weighted approach in which the DE (Differential Evolutionary algorithm) algorithm can assign suitable weights to the SVR models with most impact.
Suggestions	Univariate analysis only done. There is scope of multivariate analysis and prediction.

Research paper 3	Forecasting monthly electric energy consumption in eastern Saudi Arabia using univariate time-series analysis. Link
Algorithm / Technique / Model / Method used	Univariate Box-Jenkins time-series analysis, ARIMA.
Data set and attributes set used	Five years data. Electrical energy, temperature.
Advantages	ARIMA models require less data, have fewer coefficients, and are more accurate. The optimum ARIMA model forecasts monthly data for the evaluation year with an average percentage error of 3.8% compared to 8.1% and 5.6% for the best multiple-series regression and Abductory Induction Mechanism (AIM) models, respectively; the mean-square forecasting error is reduced with the ARIMA model by factors of 3.2 and 1.6, respectively.
Suggestions	Only temperature used as a dependent feature. There is scope of multivariate analysis and prediction.

Research paper 4	Monthly electric energy demand forecasting with neural networks and Fourier series. Link
Algorithm / Technique / Model / Method used	Economic variables usually influence the general series trend, while weather provides a periodic behavior because of its seasonal nature. This work investigates the periodic behavior of the Spanish monthly electric demand series, obtained by rejecting the trend from the consumption series. A novel hybrid approach is proposed: the periodic behavior is forecasted with a Fourier series while the trend is predicted with a Neural Network.
Data set and attributes set used	Spanish monthly electric demand from Jan1975 to Dec2002 (a total of 336 values). Past data of demand time series
Advantages	Neural networks and curve fitting are very common techniques used for time series forecasting and specifically for load forecasting. However, the combined use of neural networks and Fourier series in electric demand forecasting is a novel approach, which has been proved to provide an adequate performance: a MAPE of 1.740% has been obtained for the whole validation period, while a value of 1.326% was obtained for the best year prediction. The correct separation of trend and fluctuation and the optimization of the forecasting tools used to carry out the predictions are the key issues in the success of this technique.
Suggestions	Univariate analysis only done. There is scope of multivariate analysis and prediction.

Research paper 5	Forecasting energy consumption using ensemble ARIMA–ANFIS hybrid algorithm. Link
Algorithm / Technique / Model / Method used	ARIMA – ANFIS model.
Data set and attributes used	Iran electrical energy. Past data of demand time series.
Advantages	Special linear and nonlinear models can present desirable results with high correctness, especially when both models have good forecasting strength. ARIMA model can forecast linear part of data, ANFIS model can forecast nonlinear part, the hybrid model could increase forecasting results more efficiently. In this case the hybrid pattern that uses AdaBoost method with Genfis3 ANFIS structure and back propagation training algorithm has better results and lowest MSE value.
Suggestions / Future enhancement/ Research gap	In future studies, new diversification methods as well as using other prediction model such as Support Vector Machines can be used to improve results. Also, finding the best values of AdaBoost method parameters, with a powerful method such as genetic algorithm can make remarkable results. Moreover, each of inputs can be forecasted by ARIMA model and results used as the input of hybrid models.

6. Proposed enhancements at a very high level in this dissertation

In addition to the details mentioned in the outline the following is planned to be covered in this dissertation.

1. None of the papers above has determined the electrical energy requirements of an entire country. This dissertation aims to do so.

2. Most of the prediction is done using univariate analysis. Factors that affect the electricity consumption/generation like population, multiple weather elements (like temperature, HDD, CDD, precipitation), RDPI have not been considered in the model. However, these factors are important and we need to study the effect of these factors in the electricity consumption/generation.

7. Chapter 1 - Data

7.1. Data sources

Sl No	Data	Source	Comments
1	Monthly electricity consumption and generation in USA for fifty years. Unit = million kilo Watt Hours (mkWH)	US Energy Information Administration	Monthly energy consumption in USA for fifty years from Jan 1973 until Dec 2022 has been used.
2	Monthly USA Real Disposable Personnel Income (RDPI) for fifty years, Jan1973 until Dec2022 Unit = USD	FRED economic data	RDPI (Real Disposable Personal Income, per capita) refers to the inflation-adjusted disposable personal income per person. It represents the amount of money available to an individual or household after deducting taxes and adjusting for inflation. Per capita means that the income is divided by the total population to provide an average income per person.
3	Monthly temperature from Jan1973 until Dec2022 Unit = °F	Lincon weather and climate	Temperature reading for Lincon weather was used.
4	Monthly population from Jan1973 until Dec2022 Unit = Thousands of Persons	FRED economic data	Civilian noninstitutional population is defined as persons 16 years of age and older residing in the 50 states and the District of Columbia, who are not inmates of institutions (e.g., penal, and mental facilities, homes for the aged), and who are not on active duty in the Armed Forces. The series comes from the 'Current Population Survey (Household Survey)'
5	HDD (Heating Degree Days)	National Centers for Environmental Information	It represents the number of degrees that a day's average temperature falls below a certain base temperature (say 65°F or 18°C). It is used to estimate the energy required for heating a building during colder periods. The higher the HDD value, the colder the weather and the greater the need for heating.
6	CDD (Cooling Degree Days)	National Centers for Environmental Information	It represents the number of degrees that a day's average temperature rises above a certain base temperature (say 65°F or 18°C). It is used to estimate the energy required for cooling a building during warmer periods. The higher the CDD value, the hotter the weather and the greater the need for cooling.
7	Precipitation	National Centers for Environmental Information	Precipitation refers to any form of water, liquid or solid, that falls from the atmosphere and reaches the ground. There exists an indirect relationship between precipitation and energy consumption. Rainy or colder weather might lead to increased use of electrical appliances such as heaters, dryers, and lights. During periods of heavy precipitation or severe weather, people tend to spend more time indoors. This can lead to

			increased use of electronics, lighting, heating, and cooling systems
8	Max min temperature Unit = °F	National Centers for Environmental Information	We know that more electricity is consumed during summer for cooling and during winter for heating. However, we need to come up with a new feature called 'max-min-temperature' which will use the max temperature during the summers and min temperature during the winters. This feature gave a better correlation with the electricity consumption / electricity generation.

7.2. Data ingestion

All the weather elements i.e., HDD, CDD, minimum temperature, maximum temperature, precipitation was present in flat files / .csv files. Snapshot is as shown below.

0010271895	52.70	48.10	66.50	75.70	80.60	88.40	89.60	89.70	89.10	74.20	65.10	57.10
0010271896	53.00	59.00	63.90	80.60	87.80	87.90	91.60	94.00	89.20	75.80	68.90	57.70
0010271897	52.10	61.00	69.90	74.80	82.20	94.00	92.80	89.50	88.70	81.00	67.10	58.00
0010271898	59.10	58.00	69.90	71.00	88.00	92.70	90.50	88.40	85.40	72.30	61.00	54.30
0010271899	54.40	51.40	67.80	72.60	88.60	92.30	91.90	92.00	86.30	78.10	67.80	55.60
0010271900	55.30	56.40	64.50	76.70	84.40	85.20	90.60	93.50	89.10	78.20	66.40	57.40

Image 1: Snapshot of data from the flat file

Features	Position	Comments
State code	1-3	Range of values is 001-110 for standard states, and 111-465 for special regions.
Division number	4	Value is 0 which indicates an area- averaged element.
Element code	5-6	01 = Precipitation 02 = Average temperature 25 = Heating Degree Days 26 = Cooling Degree Days 27 = Maximum temperature 28 = Minimum temperature
Year	7-10	This is the year of record. Range is 1895 to 2022
Monthly values	11-94	11-17 Jan value, 18-24 Feb value, 25-31 Mar value 32-38 Apr value, 39-45 May value, 46-52 Jun value 53-59 Jul value, 60-66 Aug value, 67-73 Sep value 74-80 Oct value, 81-87 Nov value, 88-94 Dec value

Table 1: File format. Name, position, element description

7.3. Data pre-processing

- There were no NULL values in the data

```
# Check for null values in all columns
null_values = df_gen.isnull().sum()

# Display the count of null values for each column
print(null_values)
```

```
month         0
coal          0
petroleum     0
n_gas         0
o_gas         0
nuclear       0
p_hydel       0
c_hydel       0
wood          0
waste         0
geothermal    0
solar         0
wind          0
tot_genm      0
dtype: int64
```

```
# Check for null values in all columns
null_values = df_con.isnull().sum()

# Display the count of null values for each column
print(null_values)
```

```
month             0
residential       0
commercial        0
industrial        0
transportation    0
cust_total        0
direct            0
tot_conm          0
dtype: int64
```

Image 2: NULL values check

- Outlier detection

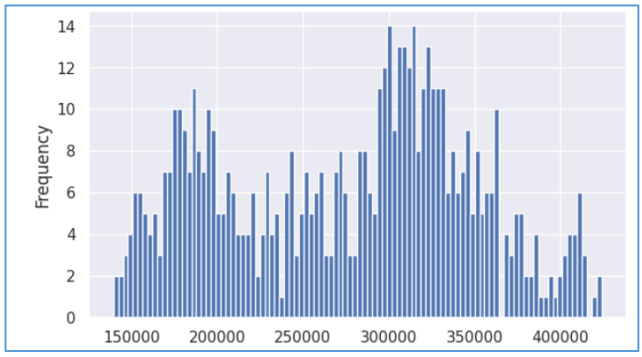


Image 3: Histogram - electrical energy generation

1. The values of electricity generation are, min value = ~150k, max value = ~430k
2. 300k to 350k is the most generated electricity value

```
df_outlier.query('elec_gen < 150000').plot(figsize = (10,4), style = '*')
df_outlier.query('elec_gen > 410000').plot(figsize = (10,4), style = '.')
```

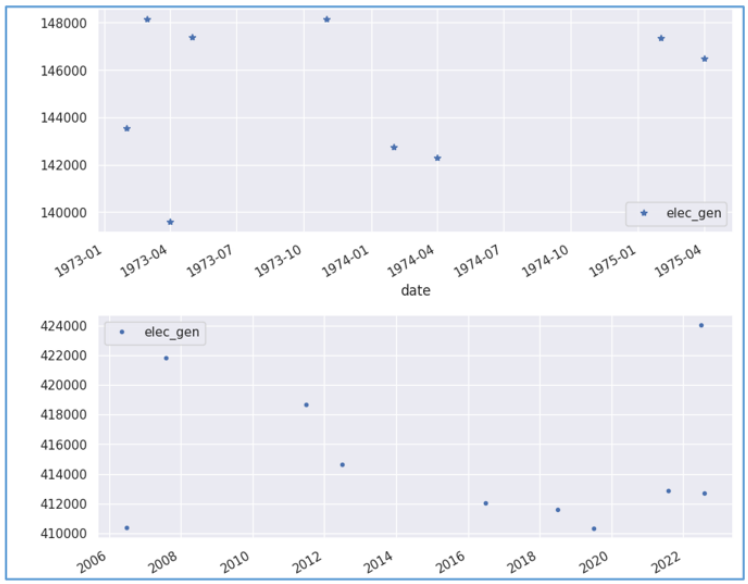


Image 4: Outliers - electrical energy generation

Here we decompose the given data and then use the residual component to check if any of the residual values exceed the threshold.

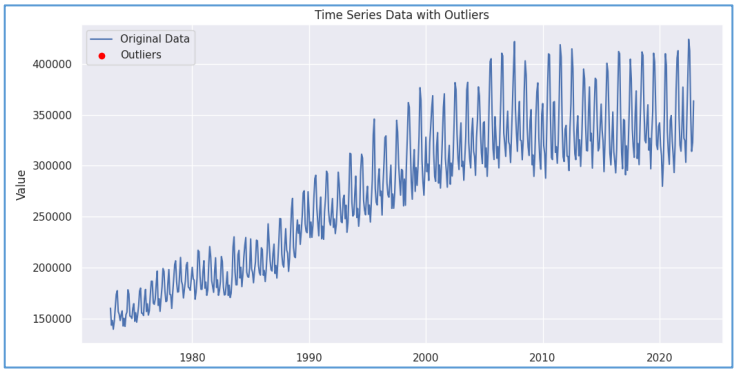


Image 5: Outliers electrical energy generation, threshold = 2.5

We do not see any outliers if we use the threshold = 2.5 (this is the normal threshold value)
However, if we decrease the threshold value to 1.05, we do see outliers.

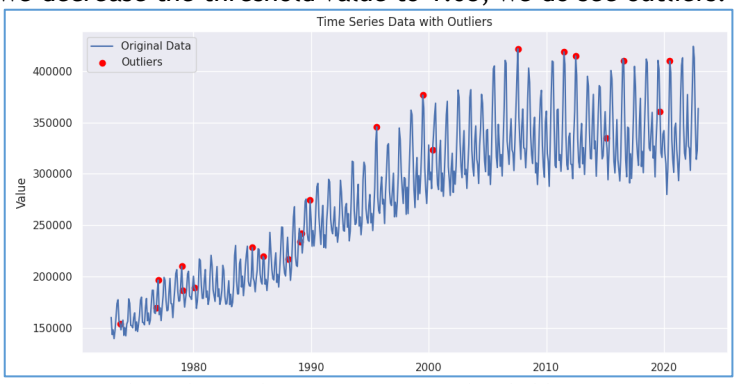


Image 6:Outliers electrical energy generation, threshold = 1.05

For the above graphs we can infer that there are no outlier values. The data is clean.

7.4. Feature extraction – months

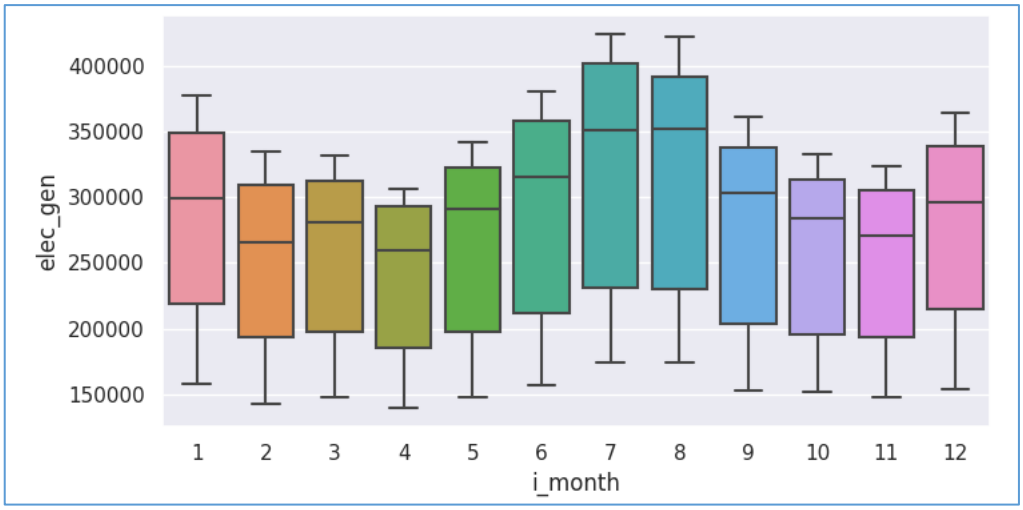
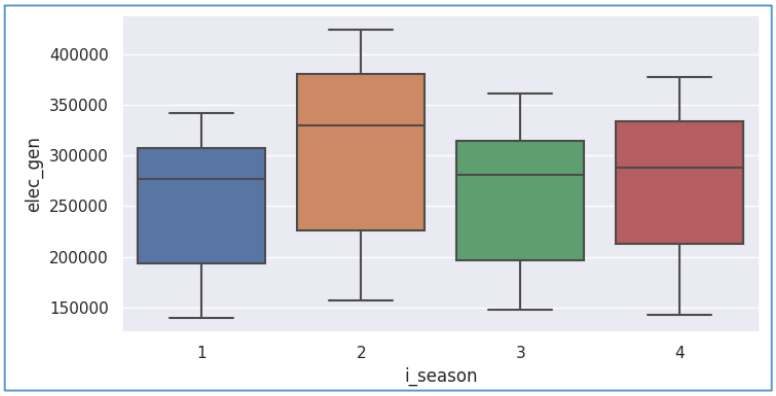


Image 7: Boxplot - Electrical energy generation Vs Months

Inference

- Every year there are 2 peaks of electricity generation, Jan, and Jul.
- Jan - heater to heat. Jul - AC to cool.
- Lowest consumption in and around Apr, Oct.
- Hence let us create feature called 'month' from the given data

7.5. Feature extraction – seasons




```
def month_season(month):
    if month in [3, 4, 5]:
        return 1 #spring
    elif month in [6, 7, 8]:
        return 2 #summer
    elif month in [9, 10, 11]:
        return 3 #fall
    else:
        return 4 #winter
```

Image 8: Electrical energy generation Vs Seasons

1-spring (3,4,5), 2-summer (6,7,8), 3-fall (9,10,11), 4-winter (12,1,2)

Inference

- Electricity consumption is max in summer (AC to cool) and then in winter (heaters to heat)
- Hence let us create the features 'seasons' and 'months' from the given time series data

```
p_value(df_ts5['elec_gen'])
```

```
P-Value :  0.45074797868417926 . It is a non-stationary data
```

The given tsd has non-stationary data. Before we do ACF, PACF calculations we need to make the tsd stationary.

```
df_as['y1'].plot()
p_value(df_as['y1'].dropna()) # need to drop the NaN values else it will give error while calculating the p_value
print('mean = ', df_as['y1'].mean().round(2), ' and variance = ', df_as['y1'].var().round(2))
```

```
P-Value :  7.782045618059567e-13 . It is a stationary data
mean =  339.58  and variance =  667150727.96
```

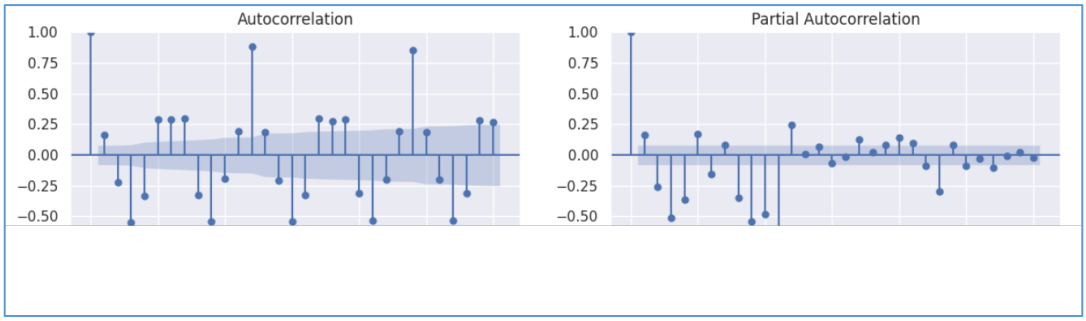


Image 9: Stationary Vs non-stationary data, ACF, PACF

Inference & next steps

- ACF gives the direct and the indirect correlation between all the lags
- PACF gives the direct correlation between the lags
- in the above graphs x-axis is the lags, y-axis is the Pearson correlation
- from the Partial Autocorrelation graph, we infer that p = 11
- from the Autocorrelation graph we infer that q = 12
- In ARIMA model for the parameters p, d, q we will use d = 1
- We will start from 1, 1, 1 and go to 12, 1, 12 to see the best values of rmse, mae etc.

7.6. Feature extraction – min- max-temperature

During the 2 hot seasons the coolers will be used (more electricity consumption) when the temperature is maximum.
During the 2 cold seasons the heaters will be used (more electricity consumption) when the temperature is minimum.
Hence, we will create a new feature called min-max-temperature (or max_min_temp) which will take the max of the temperature readings during hot seasons and the min of the temperature readings during the cold seasons.

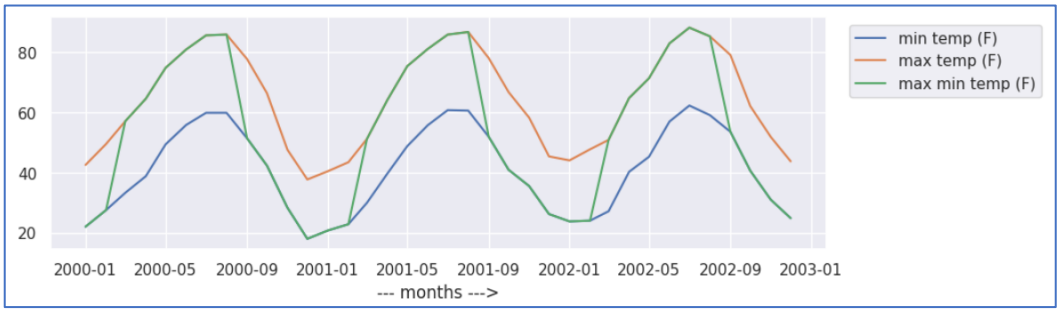


Image 10: Feature extraction, min-max temperature

Hot seasons -> 1-spring (3,4,5), 2-summer (6,7,8)
Cold seasons -> 3-fall (9,10,11), 4-winter (12,1,2)

8. Chapter 2 – Exploratory data analysis (EDA)

8.1. Data exploration and data visualization

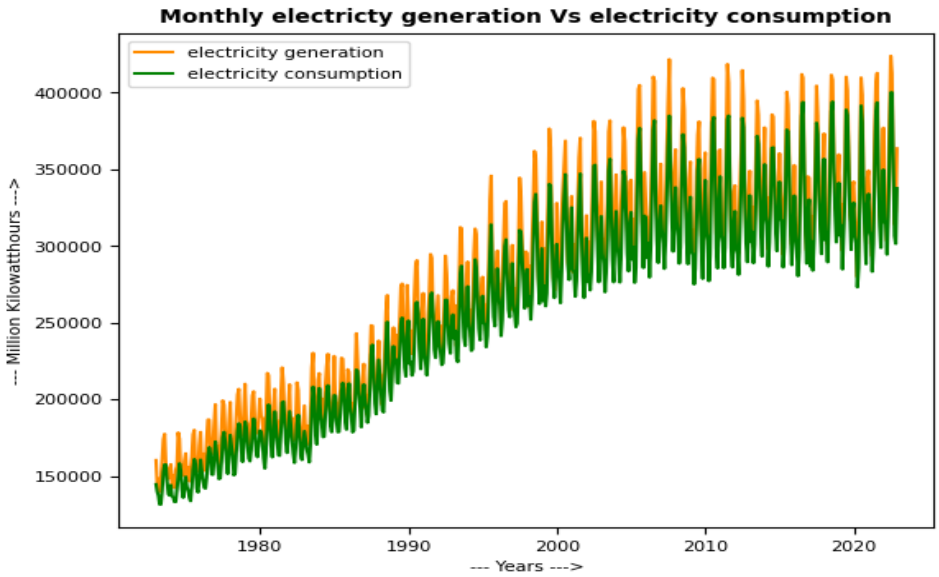


Image 11: Electricity generation Vs electricity consumption 1974-2022

Inference

1. Over the years we see that the electricity generation is always more that the electricity consumption which is what it should normally be
2. However, during a year there are several crests and troughs which need to be studied
3. Over the years the electricity consumption has increased and so has the electricity generation

Image 12: Electricity generation Vs electricity consumption, for a gap of every 8 years

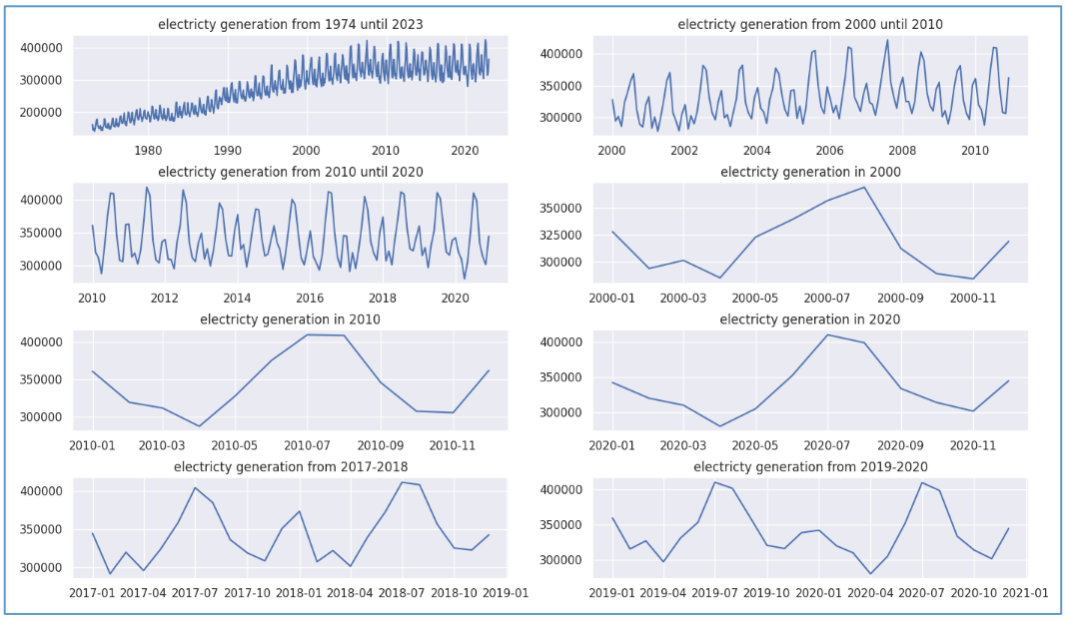


Image 13: Electricity generation for various years

Inference:

1. We see that there has been consistent more electricity generation vs electricity consumption for the 6 years 1983/91/99, 2007/15/22
2. During Sep for most of the years the electricity generation Vs electricity consumption has been neck to neck. In 1983, 1999 and in 2022 the electricity generation was less than electricity generation. This needs to be avoided.
3. Consistent spikes in generation and consumption during Jul and Dec months

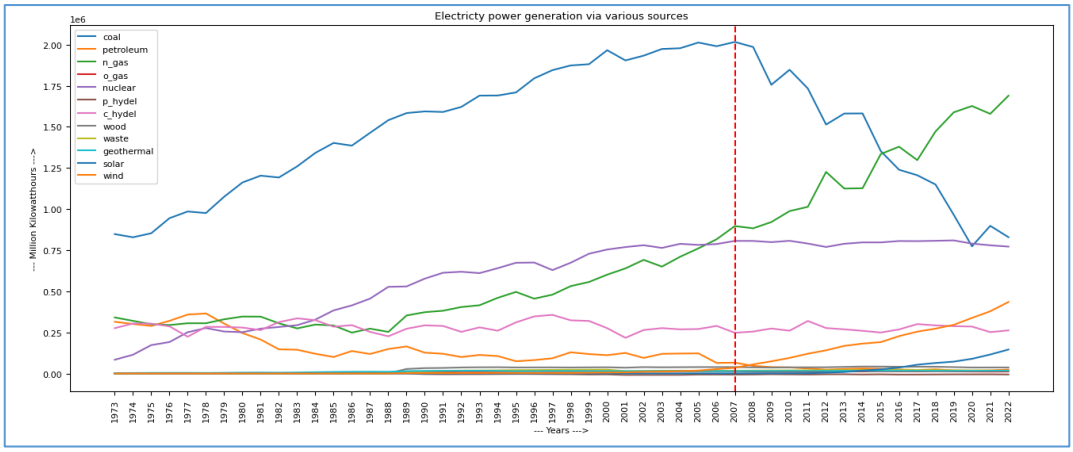


Image 14: Electricity generation from various sources 1973-2022

Inference:

1. Dependency on coal energy has been significantly reduced from 2007 when coal reached its peak
2. Dependency on natural gas energy has been considerably rising

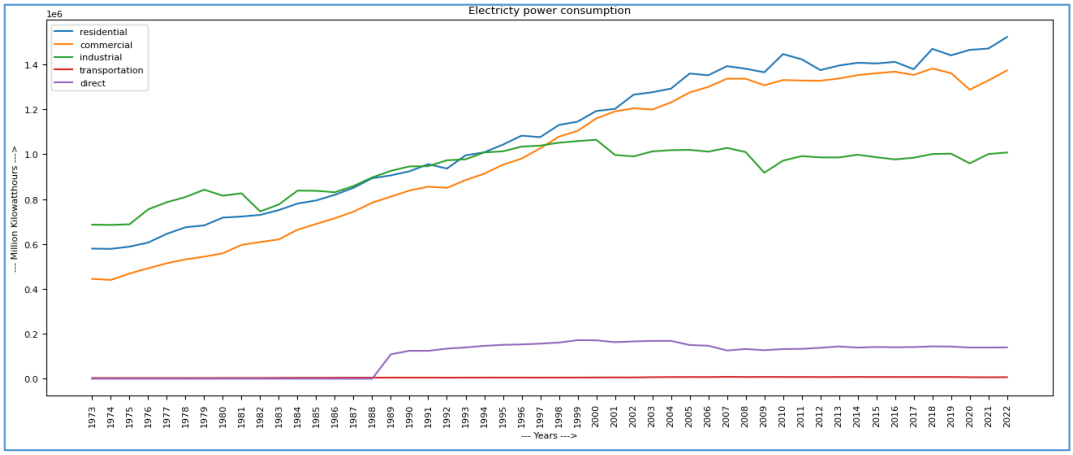


Image 15: Electricity consumption by various stakeholders

Inference:

1. Residential followed by Commercial consume most of the electrical energy produced.
2. Any deficiency in electricity production will cause maximum dis-comfort to the residents followed by discomfort and inconvenience to offices, retail stores, restaurants, hotels, hospitals, educational institutions, and other non-residential buildings.
3. Electricity consumption by the industries has been fairly remained constant since 1995.
4. Transportation uses the least amount of electrical energy when compared to other energy consumers.

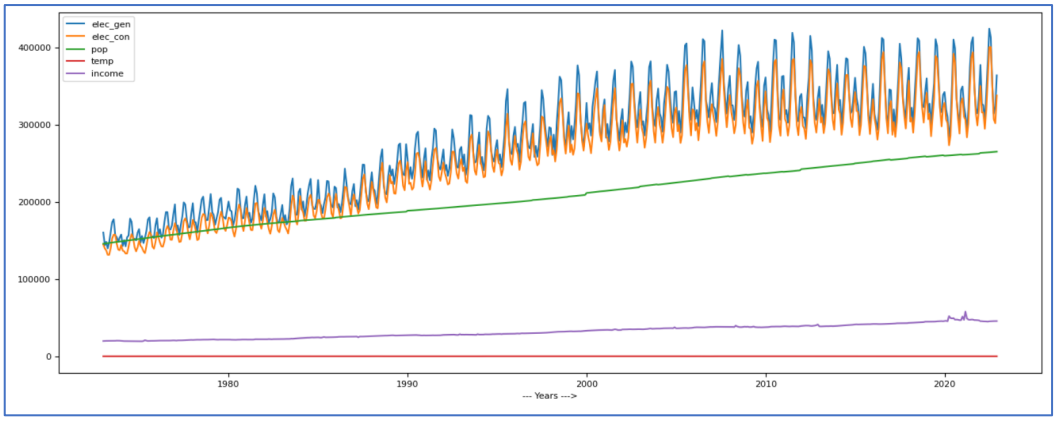


Image 16: Data scales for the various features

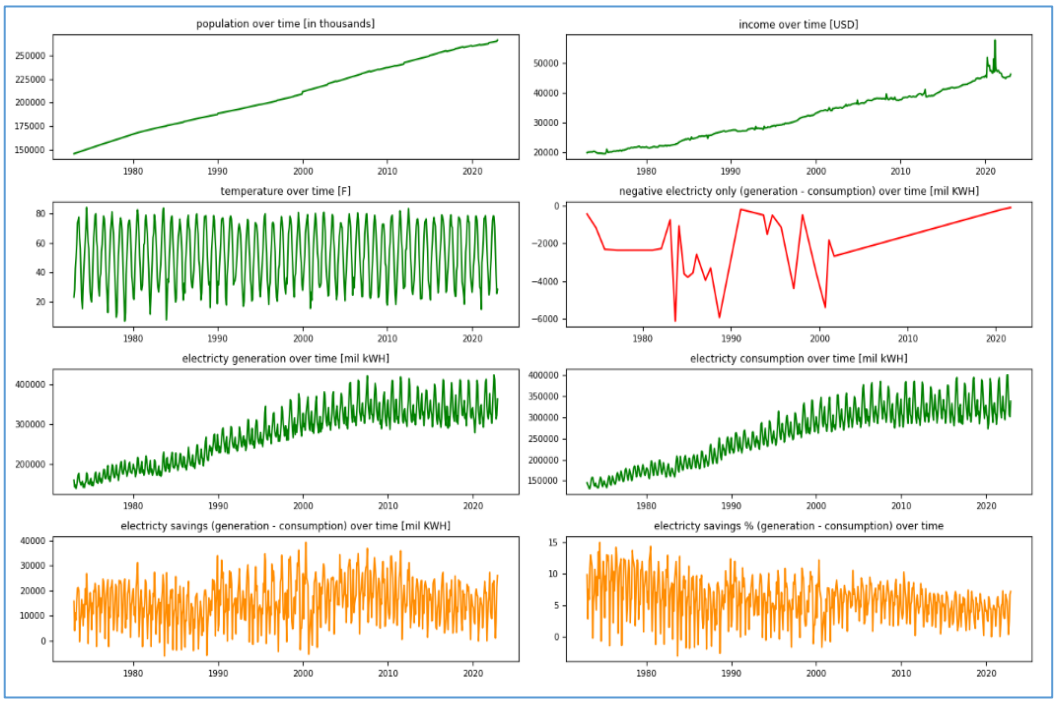


Image 17: Examining the trends over time for population, income, electricity generation / consumption

Inference

- Population over time shows a consistent rising trend
- Income over time shows a consistent rising trend, with a spike post the year 2020
- Temperature over time has been constant 0-to-80-degree Fahrenheit (-17 to around 30 degrees Celsius)
- Since 1973 there has been multiple instances wherein electricity consumption has been more than electricity generation. There have been some serious spikes around the years 1984, 1988, 2002, However, after 2002 consistent effort has been made to reduce the gap to zero
- Electricity generation and electricity consumption shows a seasonal and increasing trend over time
- Electricity savings i.e., electricity generation - electricity consumption has been good over time. Max is around 15% and min is less than -5%.

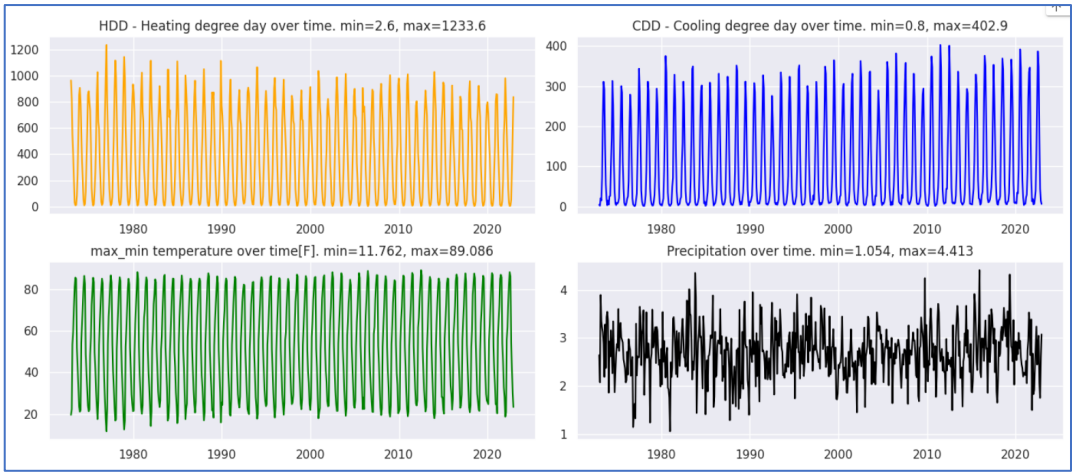


Image 18: Examining the trends over time for HDD, CDD, min-max-temperature, precipitation

Inference

- Unlike the population none of the weather elements show a rising or falling trend over time.
- There are no outliers.

8.2. Feature correlation analysis

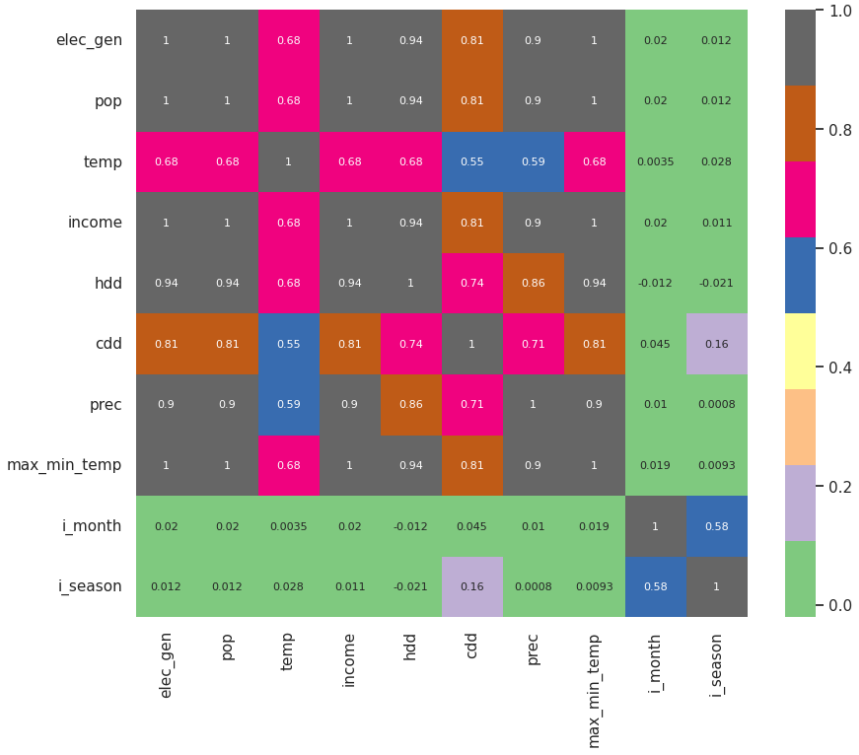


Image 19: Heat map - feature correlation

Inference

1. Electricity generation Vs Population - correlation = 1.0. Perfect positive correlation. Greater the population greater is the electricity consumption and hence electricity generation.

2. Electricity generation Vs temperature - correlation = 0.6. Positive correlation. Greater the temperature greater is the electricity consumption / generation. Data for this feature was derived from Lincoln weather and climate for a place called Lincoln in USA. The max_min_temp (via feature extraction) had a far better correlation.

3. Electricity generation Vs Income (RDPI) - correlation = 1.0. Perfect positive correlation. Greater the income greater is the is the electricity consumption and hence electricity generation.

4. Electricity generation Vs HDD - correlation = 0.94. Very high positive correlation. Higher the HDD value, the colder the weather and the greater the need for heating and hence more electricity generation required.

5. Electricity generation Vs CDD - correlation = 0.81. High positive correlation. Higher the CDD value, the hotter the weather and the greater the need for cooling and hence more electricity generation required.

6. Electricity generation Vs 'max-min-temperature' - correlation = 1.0. Perfect positive correlation. Higher the temperature more electricity generation required to cool the environment and vice versa. The average temperature across all the regions in USA or all the states in USA was taken to arrive at this feature.

7. Electricity generation Vs 'month of the year' - correlation = 0.02. Very week correlation exists between these 2 features.

8. Electricity generation Vs 'seasons of the year' - correlation = 0.012. Very week correlation exists between these 2 features.

9. Chapter 3 - Performance metrics used in the evaluation of the models

1. Mean Absolute Error (MAE) - measures the average absolute difference between the predicted values and the actual values. It provides a measure of the average magnitude of errors without considering the direction of the errors. **Lower MAE indicates better performance.**

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

2. Root Mean Squared Error (RMSE) - It is like MAE but penalizes larger errors more extremely due to the squaring operation. It provides an overall measure of the model's prediction accuracy. **Lower RMSE indicates better performance**.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

3. Mean Absolute Percentage Error (MAPE) - measures the average percentage difference between the predicted values and the actual values. It provides a relative measure of the error and is useful when we want to assess the accuracy in terms of percentage. **Lower MAPE indicates better performance.**

$$MAPE = \frac{1}{n} \times \sum \left| \frac{actual\ value - forecast\ value}{actual\ value} \right|$$

4. R-squared error of Coefficient of determination (R2) - measures the proportion of variance in the dependent variable (actual values) that is predictable from the independent variable (predicted values). **It ranges from 0 to 1. Higher values indicate better prediction performance.**

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

y_i = actual values
yi (cap) = predicted values
$\bar{y}$ = mean of the actual values

5. Mean Forecast Error (MFE) - measures the average difference between the predicted values and the actual values. **Positive MFE indicates over-prediction. Negative MFE indicates under-prediction. MFE close to zero indicates better performance.**

   MFE = (1 / n) * Σ (Forecast - Actual)

6. Best and the worst month prediction for a year – When we predict the electricity generation for a year, we will see for which months the best and worst prediction was made. For each month we will calculate the Error.

   Error = Actual – prediction
   We will then sort the errors in the ascending order and display it in a graph.

10. Chapter 4: Auto-regression, ARIMA, SARIMA, SARIMAX

10.1. Auto regression, lag (1, 2, 3 years), ML model – LR

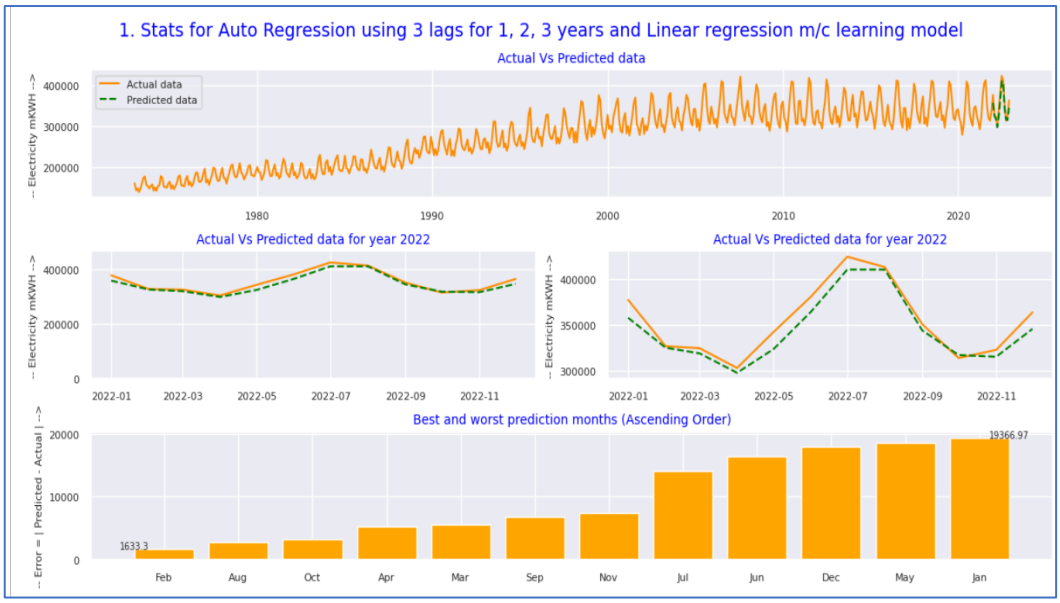


Image 20: Stats for Auto regression, ML model – Linear Regression (LR)

Used 3 different types of lags

- 1, 2, 3 months
- 3, 6, 9 months
- 12, 24, 36 months

Lags with 12, 24, 36 months gave the best performance metrics for both linear regression and random forest.

Sl No	Model description	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
1	Auto regression with lags (1,2, 3 years) with LR	9,678	12,762	2.881	0.885	-1,520	Feb	1,633	Jan	19,367

Inference

- Negative mfe shows under fitting
- r2 between 0 to 1. Higher r2 shows better prediction performance
- Lower the mae, rmse, mape values better the prediction performance
- Best month prediction = Feb, worst month prediction = Jan

10.2. Auto regression, lag (1, 2, 3 years), ML model – RF

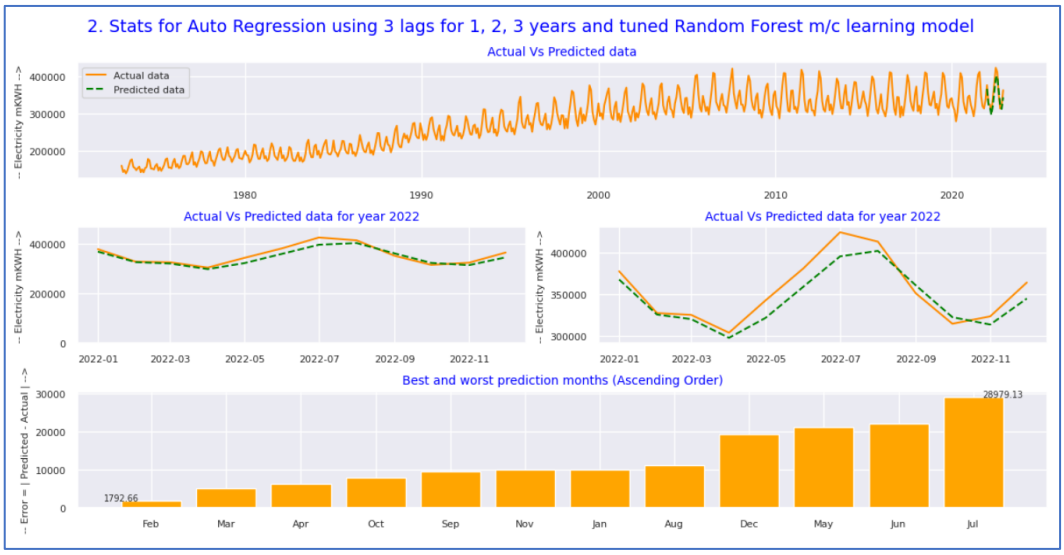


Image 21: Stats for Auto regression, ML model – Random Forest (RF)

Used the different values for the below mentioned parameters (parameter space) to get the most optimal solution

- Number of estimators = 10, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000
- Depth = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
- Max features = None, "sqrt", "auto", "log2", 0.001, 0.01, 0.5, 0.1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
- Best month prediction = Feb, worst month prediction = Jul

Number of estimators = 100, max_features = 0.01, depth = 12 gave the best performance metrics

Sl No	Model description	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
1	Auto regression with lags (1,2, 3 years) with LR	✓ 9,678	✓ 12,762	✓ 2.881	✓ 0.885	-1,520	Feb	✓ 1,633	Jan	✓ 19,367
2	Auto regression with lags (1,2, 3 years) with RF	38,518	13,568	11.117	0.870	-12	Feb	1,793	Jul	28,979

Inference

- Negative mfe shows under fitting which is better than linear regression with the same lags.
- r2 between 0 to 1. Higher r2 shows better prediction performance, in this case is slightly lesser than linear regression.
- Lower the mae, rmse, mape values better the prediction performance, all of them are comparatively higher than linear regression.

10.3. Decompose data, Auto regression, lag (1, 2, 3 years), ML model – RF

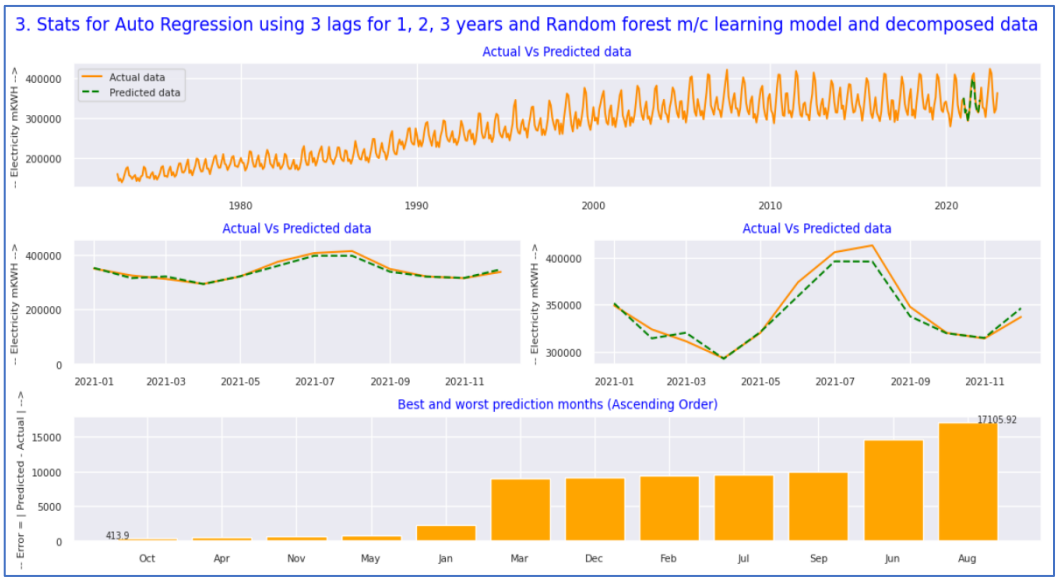


Image 22: Stats for Auto regression, decomposed data, ML model – Random Forest (RF)

We first decomposed the data into the 3 parts viz trend, seasonal, decompose using a multiplicative model. This is since the data had a seasonal and increasing trend. Each of the 3 parts were then predicted using the RF 1, 2, 3 years lag. Post that the 3 parts were then combined to give the final predicted result.

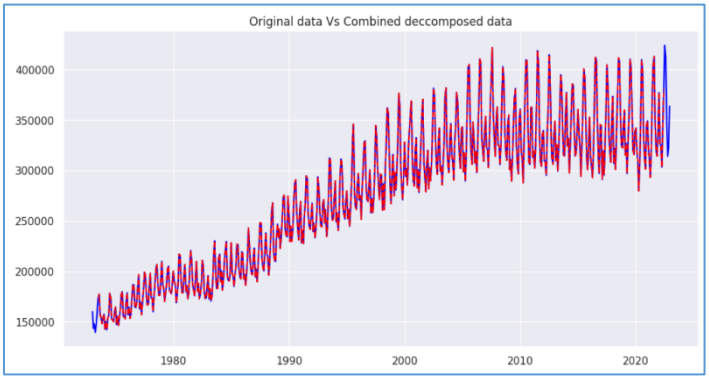


Image 23: Original Vs combined decomposed data

The decomposed values of the original data when multiplied together gives back the original value.

Inference: The predicted values of the decomposed data need to be multiplied together (not added) to arrive at the prediction.

Sl No	Model description	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
1	Auto regression with lags (1,2, 3 years) with LR	9,678	12,762	2.881	0.885	-1,520	Feb	1,633	Jan	19,367
2	Auto regression with lags (1,2, 3 years) with RF	38,518	13,568	11.117	0.870	-12	Feb	1,793	Jul	28,979
3	AR with lags (1,2, 3 years) with RF & decomposed data	✓ 7,753	✓ 9,706	✓ 2.213	✓ 0.928	3,489	Oct	✓ 414	Aug	✓ 17,106

Inference

- Positive mfe shows over fitting.
- r2 b/w 0 to 1. Higher r2 shows better prediction performance which is better than models 1 and 2.
- Lower the mae, rmse, mape values better the prediction performance which is far better than the other two models.
- Best month prediction = Oct, worst month prediction = Aug. The values have come down significantly.

10.4. Decompose data, Auto regression, lag (1, 2, 3 years), ML model – LR

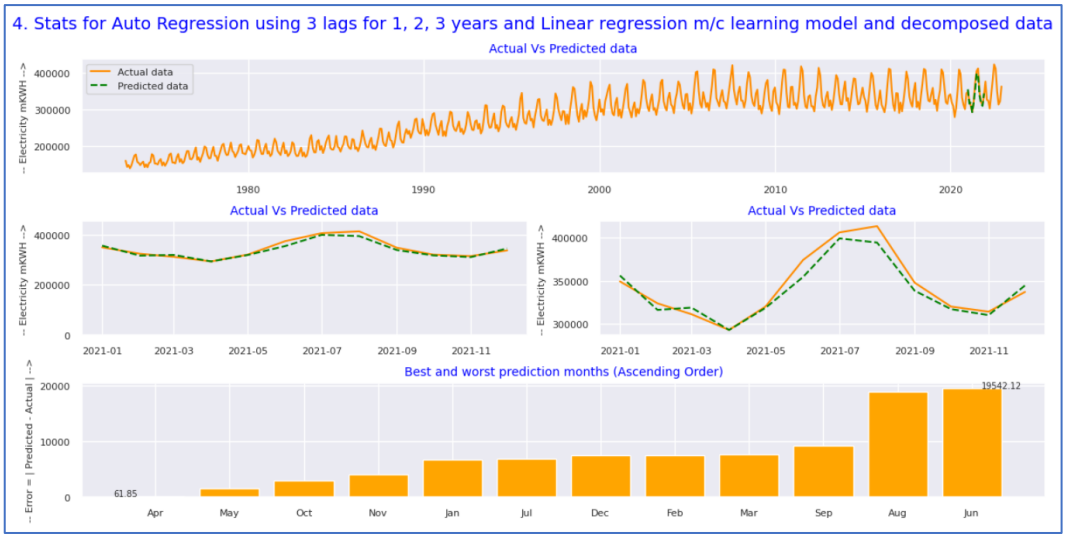


Image 24: Stats for Auto regression, decomposed data, ML model – Linear Regression (LR)

We first decomposed the data into the 3 components viz trend, seasonal, decompose using a multiplicative model. This is since the data had a seasonal and increasing trend. Each of the 3 parts were then predicted using the Linear regression 1, 2, 3 years lag. Post that the 3 parts were then combined to give the final predicted result.

Sl No	Model description	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
1	Auto regression with lags (1,2, 3 years) with LR	9,678	12,762	2.881	0.885	-1,520	Feb	1,633	Jan	19,367
2	Auto regression with lags (1,2, 3 years) with RF	38,518	13,568	11.117	0.870	-12	Feb	1,793	Jul	28,979
3	AR with lags (1,2, 3 years) with RF & decomposed data	✓ 7,753	✓ 9,706	✓ 2.213	✓ 0.928	3,489	Oct	414	Aug	✓ 17,106
4	AR with lags (1,2, 3 years) with LR & decomposed data	9,350	11,946	2.779	0.891	-1,302	Apr	✓ 62	Jun	19,542

Inference

- Negative mfe shows under fitting
- r2 b/w 0 to 1. Higher r2 shows better prediction performance which is second best.
- Lower the mae, rmse, mape values better the prediction performance which is again second best.
- Best month prediction = Apr, worst month prediction = Jun. The values for the best prediction have come down significantly to double digits

10.5. ARIMA model for prediction

ARIMA (Auto Regressive Integrated Moving Average) has 3 parameters.
p - Order of Autoregression (AR): The autoregressive order is the number of past observations (lags) that are used as predictors for the current observation. It captures the relationship between the current observation and its previous values. Higher value of p indicates a stronger dependency on past observations.

d - Order of Differencing: The differencing order is the number of times the tsd (time-series-data) is differenced to make it stationary. Differencing removes the trend from the tsd, making it more suitable for ARMA modeling. d = 0 if the tsd is already stationary.

q - Order of Moving Average (MA): The moving average order is the number of lagged forecast errors that are used as predictors for the current observation. It captures the relationship between the current observation and past forecast errors. Higher value of q indicates a stronger dependency on past forecast errors.

We will do fine tuning of these 3 parameters p, d, q

- Step 1: take d = 1 to 12 and fix the ones which give the best scores
- Step 2: for the best d; take p = 1 to 12 and fix the ones which give the best scores
- Step 3: for the best d, p; take q = 1 to 12 and fix the ones which give the best scores
- Step 4: now we have got the best values of p, d, q

Note: in each of the steps 1 to 3 when we do the prediction if we convergence coming as 'False' then we skip the prediction step for those parameters and go to the next step. Convergence being 'False' signifies that the model parameters used to iteratively optimize the solution is not able to trend to a solution.

After using the above permutation and combinations we get the best performance scores with the following three combinations of p, d, q.
(p, d, q) = (10, 2, 1), (9, 2, 1), (10, 1, 1)

Sl No	Model details	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
1	Auto regression with lags (1,2, 3 years) with linear regression	9,678	12,762	2.881	0.885	-1,520	Feb	1,633	Jan	✓ 19,367
2	Auto regression with lags (1,2, 3 years) with random forest	38,518	13,568	11.117	0.870	-12	Feb	1,793	Jul	28,979
3	Auto regression with lags (1,2, 3 years) with random forest & decomposed data	✓ 7,753	✓ 9,706	✓ 2.213	✓ 0.928	✓ 3,489	Oct	✓ 414	Aug	✓ 17,106
4	Auto regression with lags (1,2, 3 years) with linear regression & decomposed data	✓ 9,350	11,946	2.779	0.891	-1,302	Apr	✓ 62	Jun	✓ 19,542
5a	ARIMA model (10, 2, 1)	12,006	14,568	3.258	0.846	9,096	Sep	773	Jan	23,868
5b	ARIMA model (09, 2, 1)	13,690	16,697	3.706	0.798	6,677	Mar	3,422	Jul	33,115
5c	ARIMA model (10, 1, 1)	13,763	16,293	3.763	0.807	10,759	Nov	798	Jan	26,713
	Tick marks is	x < = 5%	x < = 5%	x < = 5%	x > = 75%	25%<x< 50%		x < = 20%		x < = 20%

Inference

- Positive mfe shows over fitting for all the 3 tuned ARIMA models
- r2 between 0 to 1. Higher r2 shows better prediction performance, which is best for (10, 2, 1). However, this is the least good against all the other four models
- Lower the mae, rmse, mape values better the prediction performance – mae, mape for (10, 2, 1) ranks fourth against all the 5 models, rmse is far behind all the four models
- Best month prediction = Sep for (10, 2, 1) is third in rank (against all the five models), worst month prediction = Jan is 4th in rank (against all the five models).

When we applied the ARIMA model on the decomposed data, the results were inconclusive (convergence was coming as 'False'). Hence, the stats are ignored for this case.

10.6. SARIMA, SARIMAX models for prediction

SARIMAX (Seasonal Auto Regressive Integrated Moving Average with eXogneous variables) has 7 parameters. (p, d, q, P, D, Q, S)
P – Seasonal Auto Regressive order (AR): It is the number of past observations (lags) for the seasonal component of the tsd that are used as predictors for the current observation.
It indicates how many lagged values of the seasonal series need to be included in the model.
For example, P = 2 implies use the last two seasonal data for current prediction.

D – Seasonal order of differencing: It is the number of seasonal differences required to make the tsd stationary (remove seasonality from the data).

Q – Seasonal Moving Average order (MA): It is the number of MA terms for the seasonal component of the tsd that are used as predictors for the current observation.
It indicates how many lagged forecast errors of the seasonal series need to be included in the model.
For example, Q = 1 implies use the last forecast error of the seasonal series for current prediction.

S – Seasonal Periodicity: It is the seasonal periodicity of the tsd or the number of time periods in a seasonal cycle.
For example, S = 12 implies the tsd has a monthly data and it shows seasonality over a year S=12.

The following studies were done in phases using the 'auto_arima' function for the SARIMA model.

Please read the parameters as ARIMA (p, d, q) (P, D, Q), S = 12 for all the studies of phases.

<table>
<tr><th>SARIMA model phase number</th><th>Parameters</th><th>Best model (p, d, q) (P, D, Q)
Total fit time (seconds)</th></tr>
<tr><td>1</td><td>stepwise = False</td><td>(0,1,1) (2,0,0) [12]
270.665 seconds</td></tr>
<tr><td>2</td><td>stepwise = True</td><td>(0,1,1) (2,0,0) [12]
61.370 seconds</td></tr>
<tr><td colspan="3">Inference
a) "stepwise = True" gave results in less steps and less time than when compared to "stepwise = False"
b) Best parameters are same. (0,1,1) (2,0,0) [12]
Hence, use stepwise = True going forward</td></tr>
<tr><td>3</td><td>start_p=0, default value of start_p =1
max_p=3, max_q=3,
d=None, D=1, stepwise = True</td><td>(3,0,0) (0,1,1) [12]
103.673 seconds</td></tr>
<tr><td>4</td><td>start_p=0, start_q=0, start_P=0, start_Q=0,
max_p=12, max_q=12, max_P=12, max_Q=12
d=None, D=1, stepwise = True</td><td>(3,0,0) (0,1,1) [12]
67.929 seconds</td></tr>
<tr><td colspan="3">Inference - From Phase 3 and 4 we infer that
a) if we specify the starting values of p, q, P, Q (Phase 4) as zero the results arrive faster than when no start values are provided (Phase 3). In the latter case we allow the model to choose its own default values and hence it takes comparatively more time.
b) even though in Phase 4 the max values of p, q, P, Q was 12 then compared to Phase 3 where max values of p, q was 3. Phase 4 converged to the same results faster
c) Best parameters are same. (3,0,0) (0,1,1) [12]</td></tr>
<tr><td>5</td><td>start_p=0, start_q=0, start_P=0, start_Q=0,
max_p=12, max_q=12, max_P=12, max_Q=12
d=1, D=1, stepwise = True</td><td>(0,1,2) (2,1,2) [12]
916.816 seconds</td></tr>
<tr><td>6</td><td>start_p=0, start_q=0, start_P=0, start_Q=0,
max_p=12, max_q=12, max_P=12, max_Q=12
d=1, D=None, stepwise = True</td><td>(0,1,1) (2,0,0) [12]
81.950 seconds</td></tr>
</table>

7	start_p=0, start_q=0, start_P=0, start_Q=0, max_p=12, max_q=12, max_P=12, max_Q=12 d=None, D=None, stepwise = True	(4,0,0) (1,0,1) [12] 318.495 seconds

From Phase 4, 5, 6, 7 we infer that for the same min, max values of p, q, P, Q (0,12)

a) Least time to arrive at the best values was for phase 4 when d=None, D=1
b) Best parameter values are
   - Phase 4, d=None, D=1, (3,0,0) (0,1,1) [12]
   - Phase 5, d=1, D=1, (0,1,2) (2,1,2) [12]
   - Phase 6, d=1, D=None, (0,1,1) (2,0,0) [12]
   - Phase 7, d=None, D=None, (4,0,0) (1,0,1) [12]

Table 2: Best model parameters and total fit time for the various SARIMA models

We need to find for which set of values we get the best performance metrics (mae, rmse, mape, r2, mfe)

- Phase 1,2, (0,1,1) (2,0,0) [12]
- Phase 3,4, (3,0,0) (0,1,1) [12], d=None, D=1
- Phase 5, (0,1,2) (2,1,2) [12], d=1, D=1
- Phase 6, (0,1,1) (2,0,0) [12], d=1, D=None
- Phase 7, (4,0,0) (1,0,1) [12], d=None, D=None

After running the SARIMA function for the above parameters (p, d, q, P, D, Q, S=12) we get

SARIMA model + parameter values (p, d, q, P, D, Q, S)	mae	rmse	mape	R2	mfe
SARIMA phase 1,2: 0,1,1,2,0,0,12	41,842	17,224	11.515	0.785	14,347
SARIMA phase 3,4: 3,0,0,0,1,1,12	42,258	16,305	11.652	0.807	13,494
SARIMA phase 5: 0,1,2,2,1,2,12	✓ 41,186	✓ 14,262	11.374	✓ 0.853	✓ 11,621
SARIMA phase 6: 0,1,1,2,0,0,12	41,842	17,224	11.515	0.785	14,347
SARIMA phase 7: 4,0,0,1,0,1,12	✓ 40,849	17,371	✓ 11.215	0.781	14,984
Tick marks is	x < = 25%	x < = 25%	x < = 25%	x > = 75%	x < = 25%

From the above performance statistics, it is evident the best performance scores come for 0, 1, 2, 2, 1, 2, 12
Even though the mae, mape score for phase 5 is comparatively higher then compared to phase 7, the rmse is better. R2 is the best and so is mfe. Positive mfe shows a bit of over fitting.

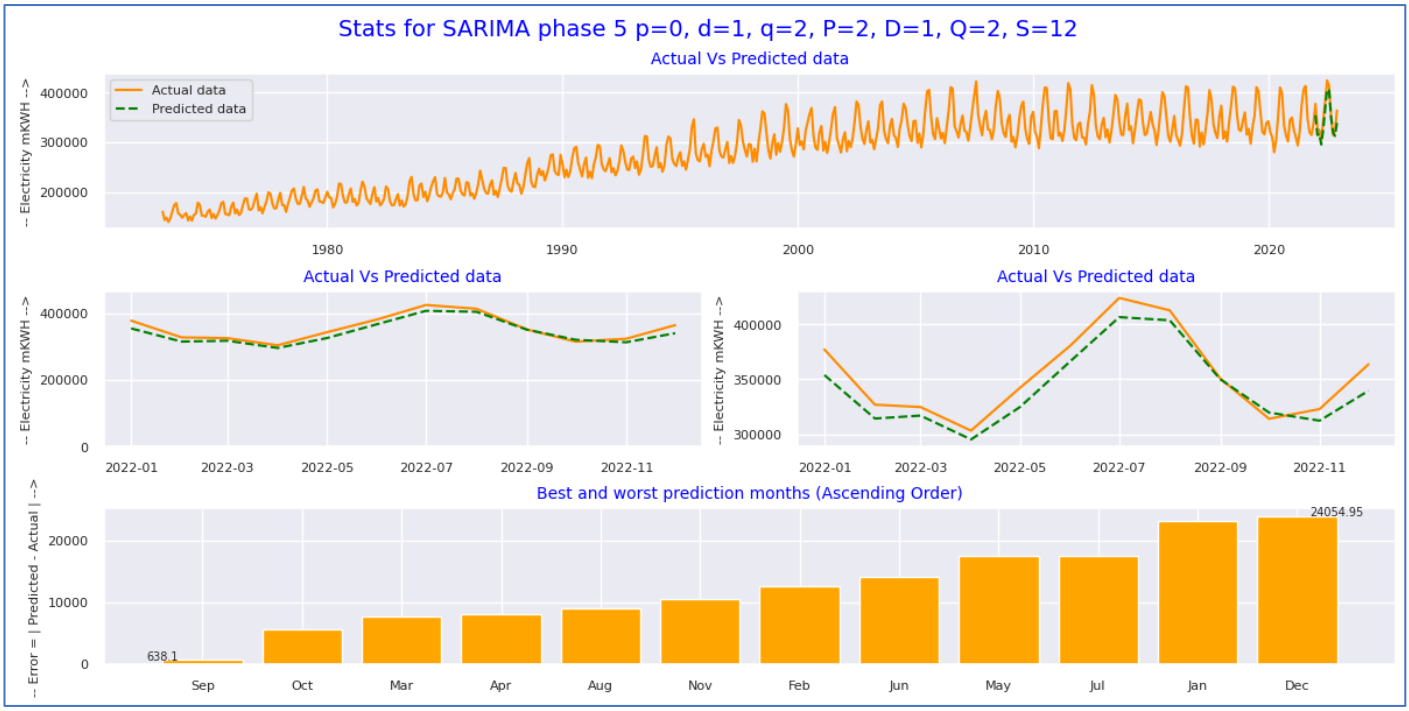


Image 25: Stats for SARIMA model

Performance of the various models										
Sl No	Model details	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
1	Auto regression with lags (1,2, 3 years) with linear regression	9,678	12,762	2.881	0.885	-1,520	Feb	1,633	Jan	19,367
2	Auto regression with lags (1,2, 3 years) with random forest	38,518	13,568	11.117	0.870	-12	Feb	1,793	Jul	28,979
3	Auto regression with lags (1,2, 3 years) with random forest & decomposed data	✓ 7,753	✓ 9,706	✓ 2.213	✓ 0.928	3,489	Oct	414	Aug	✓ 17,106
4	Auto regression with lags (1,2, 3 years) with linear regression & decomposed data	✓ 9,350	11,946	2.779	0.891	-1,302	Apr	✓ 62	Jun	19,542
5a	ARIMA model (10, 2, 1)	12,006	14,568	3.258	0.846	9,096	Sep	773	Jan	23,868
5b	ARIMA model (09, 2, 1)	13,690	16,697	3.706	0.798	6,677	Mar	3,422	Jul	33,115
5c	ARIMA model (10, 1, 1)	13,763	16,293	3.763	0.807	10,759	Nov	798	Jan	26,713
6a	SARIMA model (0, 1, 1, 2, 0, 0, 12)	41,842	17,224	11.515	0.785	14,347	Feb	3,577	Jan	30,957
6b	SARIMA model (1, 1, 1, 2, 0, 0, 12)	39,627	16,208	10.904	0.809	11,968	Feb	1,062	Jan	33,432
6c	SARIMA model (0, 1, 1, 2, 1, 2, 12)	41,186	14,262	11.374	0.853	11,621	Sep	638	Dec	24,055
	Tick marks is	x < = 5%	x < = 5%	x < = 5%	x > = 95%	15%<x< 30%		x < = 5%		x < = 5%

Inference

- Positive mfe shows over fitting for all the 3 SARIMA models
- r2 between 0 to 1. Higher r2 shows better prediction performance, which is not the best when compared across all the models, except when you compare 6c with the ARIMA models
- Lower the mae, rmse, mape values better the prediction performance –
  - mae, mape, mfe is worse than compared to all the 5 models
  - rmse, r2 score for 6c is better than ARIMA but worse than others
- 6c is the first model so far which has predicted best for Sep where we had electricity consumption more than electricity generation
- Best model show far is under Sl No 3

The following studies were done in phases using the 'auto_arima' function for the SARIMAX model with exogeneous features.

Please read the parameters as ARIMA (p, d, q) (P, D, Q), S = 12 for all the studies of phases.

The 8 independent features are 'pop', 'income', 'hdd', 'cdd', 'prec', 'max_min_temp', 'i_month', 'i_season'. ['prec' stands for precipitation]

When we use the auto_arima model to find the best values of the parameters using all the 8 features as exogeneous features we get the parameters as (3,0,2) (1,0,1) [12].
It took a total fit time of 775.954 seconds with a range of 0 to 3.

However, the model failed while building with these 8 features. Hence, we use only 'pop', 'income' and 'max_min_temp' to start with. These were the 3 features with 100% correlation with electricity generation.

For SARIMAX models in phase 12(3 features), 13(4 features) we go the same parameters value (0, 1, 2, 2, 1, 2, 12).

Hence, for models in phase 14(5 features) and 15(6 features) we used the same parameter values.
Refer SARIMAX phases 8 to 15.

Performance of the SARIMA, SARIMAX models					
SARIMA model + parameter values (p, d, q, P, D, Q, S)	mae	rmse	mape	R2	mfe
SARIMA phase 1,2: 0,1,1,2,0,0,12	41,842	17,224	11.515	0.785	14,347
SARIMA phase 3,4: 3,0,0,0,1,1,12	42,258	16,305	11.652	0.807	13,494
SARIMA phase 5: 0,1,2,2,1,2,12	41,186	14,262	11.374	0.853	11,621
SARIMA phase 6: 0,1,1,2,0,0,12	41,842	17,224	11.515	0.785	14,347
SARIMA phase 7: 4,0,0,1,0,1,12	40,849	17,371	11.215	0.781	14,984
SARIMAX phase 8: Features - population, max_min_temp, income. Range(0-2). d=None, D=None	✓ 39,795	14,524	✓ 10.989	0.847	9,734
SARIMAX phase 9: Features - population, max_min_temp, income. Range(3-5). d=None, D=None	40,487	14,742	11.170	0.842	11,286
SARIMAX phase 10: Features - population, max_min_temp, income. Range(0-3). d=1, D=None	40,348	10,017	11.338	0.927	✓ 746
SARIMAX phase 11: Features - population, max_min_temp, income. Range(0-3). d=None, D=1	40,337	10,309	11.214	0.923	6,613
SARIMAX phase 12: 3 Features - population, max_min_temp, income. Range(0-3). d=1, D=1	✓ 39,774	9,320	✓ 11.088	0.937	4,678
SARIMAX phase 13: 4 Features - population, max_min_temp, income, HDD. Range(0-3). d=1, D=1	40,877	8,977	11.415	0.942	4,285
SARIMAX phase 14: 5 Features - population, max_min_temp, income, HDD, precipitation. Range(0-3). d=1, D=1	40,921	8,898	11.424	0.943	4,475
SARIMAX phase 15: 6 Features - population, max_min_temp, income, HDD, precipitation, CDD. Range(0-3). d=1, D=1	41,235	✓ 6,338	11.535	✓ 0.971	✓ 2,599
Tick marks is	x < = 15%	x < = 15%	x < = 15%	x > = 95%	x < = 15%

We get the best scores for d=1, D=1 while using the 6 features 'population', 'income', 'max_min_temp', 'HDD', 'precipitation', 'CDD'.

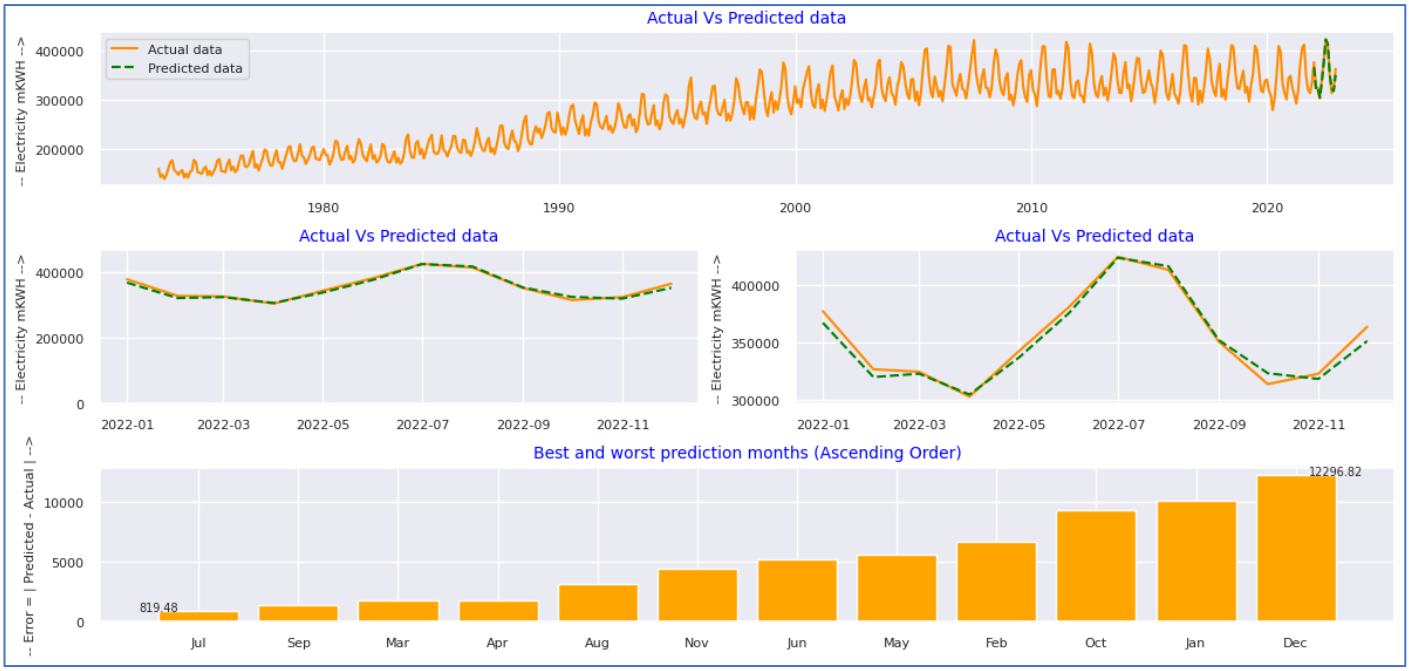


Image 26: Stats for SARIMAX model

When we compare this SARIMAX model with all the best models achieved so far, we get the following.

Performance of the various models											
Sl No	Model details	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
1	Auto regression with lags (1,2, 3 years) with linear regression	9,678	12,762	2.881	0.885	-1,520	Feb	1,633	Jan	19,367
2	Auto regression with lags (1,2, 3 years) with random forest	38,518	13,568	11.117	0.870	-12	Feb	1,793	Jul	28,979
3	Auto regression with lags (1,2, 3 years) with random forest & decomposed data	✓ 7,753	9,706	✓ 2.213	✓ 0.928	✓ 3,489	Oct	✓ 414	Aug	17,106
4	Auto regression with lags (1,2, 3 years) with linear regression & decomposed data	✓ 9,350	11,946	2.779	0.891	-1,302	Apr	✓ 62	Jun	19,542
5a	ARIMA model (10, 2, 1)	12,006	14,568	3.258	0.846	9,096	Sep	773	Jan	23,868
5b	ARIMA model (09, 2, 1)	13,690	16,697	3.706	0.798	6,677	Mar	3,422	Jul	33,115
5c	ARIMA model (10, 1, 1)	13,763	16,293	3.763	0.807	10,759	Nov	798	Jan	26,713
6a	SARIMA model (0, 1, 1, 2, 0, 0, 12)	41,842	17,224	11.515	0.785	14,347	Feb	3,577	Jan	30,957
6b	SARIMA model (1, 1, 1, 2, 0, 0, 12)	39,627	16,208	10.904	0.809	11,968	Feb	1,062	Jan	33,432
6c	SARIMA model (0, 1, 1, 2, 1, 2, 12)	41,186	14,262	11.374	0.853	11,621	Sep	✓ 638	Dec	24,055
7.1	SARIMAX model, 3 exogenous features	39,774	9,320	11.088	✓ 0.937	✓ 4,678	Mar	✓ 322	Dec	✓ 15,648
7.2	SARIMAX phase 15: 6 exogeneous features	41,235	✓ 6,338	11.535	✓ 0.971	✓ 2,599	Jul	820	Dec	✓ 12,297

Inference

- The mae, mape score of SARIMAX models 7.1, 7.2 is significantly higher when compare to model 3 and hence performance is worse.
- The rmse score for 7.1, 7,2 is comparatively lower than model 3 and hence performance is better.
- The R2 score for 7.1, 7,2 is comparatively higher than model 3 and hence performance is better.
- SARIMAX model is better than compared to the other SARIMA models.
- The best and worst months are Mar, Dec respectively. These scores are better than the model 3.
- We can say that models 3, 7.1, 7.2 has a tie so far. However, model at Sl No 3 is best.

11. Chapter 5: XGBoost (eXtreme Gradient Boost)

XGBoost (eXtreme Gradient Boosting) is a powerful ML algorithm. It is used for classification as well as regression tasks. It is an ensemble learning method i.e., it combines the predictions from multiple ML models to create a more robust and accurate predictive model. By taking an aggregate of the outputs of several models, the ensemble can often achieve better performance than the individual model itself.

11.1. XGBoost model 1.1, 1.2, 1.3 – The first stab

Features used - month, season

- For the 1st set of XGBoost models we via feature engineering extracted the month from the date. This was used as an independent feature, since the electricity generation was dependent on months. Refer here.
- From months we extracted seasons. We found that the electric generation was dependent on seasons too. Refer here.

Inference

- Using the below mentioned parameters and its value the rmse had a very dismal score.
- We did not calculate any other performance metrics for these models.

			Parameters and its value						
Model #	Features used	XGBoost Model #	objective: reg	early_stopping_rounds	enable_categorical	learning_rate	missing	n_estimators	rmse
1.1	month, season	1.1	squarederror		FALSE		nan	1,000	81,128
1.2	month, season	1.2	squarederror	50	FALSE	0.001	nan	1,000	1,83,021
1.2	month, season	1.3	squarederror	50	FALSE	0.0001	nan	1,000	3,29,593

Note: for all XGBoost models in this artifact we used the following parameters and its values. Hence, they will not be mentioned explicitly.

i. objective: reg = squarederror
ii. enable_categorical = False
iii. missing = nan

11.2. XGBoost model 2.1, 2.2 – More features

It is evident that we need more features for prediction. Hence, we used the following features for the 2nd XGBoost model.

Features used – month, season, population, income, HDD, CDD, precipitation, max-min-temperature

Difference between model 2.1 and 2.2 is the max_depth value.

XGBoost Model #	Parameters and its value													mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
	base_score	booster	early_stopping_rounds	min_child_weight	subsample	colsample_bylevel	colsample_bytree	scale_pos_weight	reg_lambda	reg_alpha	learning_rate	max_depth	n_estimators									
2.1	0.5	gbtree	50								0.01	3	1,000	8,649	10,230	2.438	0.924	8,536	Oct	679	Dec	18,258
2.2	0.5	gbtree	50								0.01		1,000	7,498	8,890	2.108	0.943	7,294	Apr	328	Dec	16,397

Inference

The performance scores are better without using the max_depth = 3 value. Hence, we will stop using max_depth = 3.

We see that the top 3 features out of all the 8 features used were income, population and CDD.
Hence, we produce an XGBoost model 3 with these 3 features only.

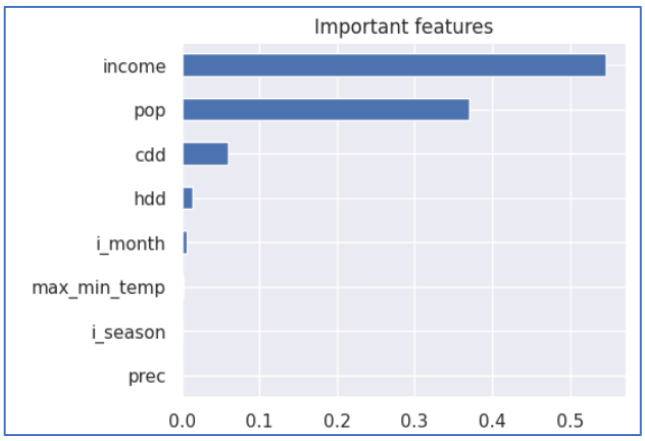


Image 27: XGBoost - Important features

11.3. XGBoost model 3 – Important features from XGBoost

Features used - Income, population and CDD

XGBoost Model #	Parameters and its value													mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
	base_score	booster	early_stopping_rounds	min_child_weight	subsample	colsample_bylevel	colsample_bytree	scale_pos_weight	reg_lambda	reg_alpha	learning_rate	max_depth	n_estimators									
2.1	0.5	gbtree	50								0.01	3	1,000	8,649	10,230	2.438	0.924	8,536	Oct	679	Dec	18,258
2.2	0.5	gbtree	50								0.01		1,000	7,498	8,890	2.108	0.943	7,294	Apr	328	Dec	16,397
3	0.5	gbtree	50								0.01		1,000	8,503	11,192	2.455	0.909	✓ 46	May	931	Jan	20,253

Inference

- The 3rd model has the best mfe score. This shows that the over fitting is the least amongst the other models.
- However, the performance scores decreases if we use only the top 3 independent features given by the XGBoost model.
- Hence, going forward we will use all the features and not just the 3 features.

11.4. XGBoost model 4.1 to 4.7 – Cross validation

For the XGBoost model 4 we will use Cross Validation.

We have monthly data from 1973 until 2022 i.e., for 50 years.
We will do cross validation for 5 years.
Hence, number of splits = 5, test size = for 1 year = 12 months since we have monthly data, gap = 12 months.

XGBoost Model 4.1

We split the data in 5 folds.

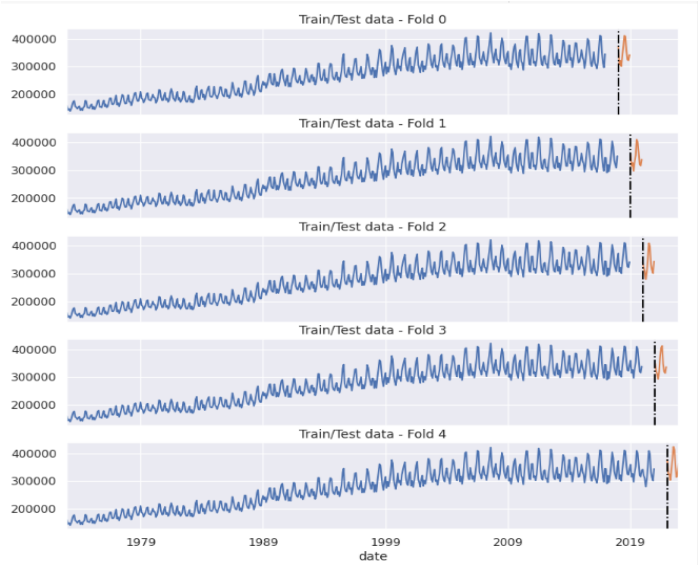


Image 28: Data in five folds

Fold No	Train data year, length	Gap year	Test data year
1	1973 to 2016, 528	2017	2018
2	1973 to 2017, 540	2018	2019
3	1973 to 2018, 552	2019	2020
4	1973 to 2019, 564	2020	2021
5	1973 to 2020, 576	2021	2022

Features used – month, season, population, income, max-min-temperature, lag1, lag2, lag3.

Here lag1, lag2, lag3 are the lagged values of the target feature 'elec_gen' by 1, 2, 3, years respectively.

For each fold we do the prediction. The prediction for the final year is the mean prediction of all the 5 folds. Similarly, the final rmse score is the average rmse score of the individual 5 folds.

Fold number	Prediction for that fold	rmse
1	357668.22, 313284.88, 311380.75, 298523.60, 325156.47, 370843.90, 398659.50, 399179.06, 343623.84, 312291.66, 309411.20, 336375.56	10753.42
2	360845.94, 321950.00, 321029.00, 305744.00, 326059.90, 360219.90, 402436.38, 395921.56, 357017.06, 319675.25, 319732.90, 335147.16	5514.65
3	351441.25, 306997.53, 312928.78, 296076.10, 321296.12, 365232.97, 389879.53, 388279.03, 340313.60, 314170.22, 307938.20, 329043.3	12259.98
4	345924.66, 328288.78, 317005.16, 293783.80, 315941.10, 378611.62, 402249.53, 408558.84, 349901.22, 315935.88, 307774.03, 343506.34	4461.03
5	361859.00, 319110.94, 313919.94, 288582.10, 323231.00, 364545.12, 403671.75, 406028.78, 358274.97, 317019.28, 308653.10, 338927.30	14682.65
Average	355547.81, 317926.43, 315252.73, 296541.92, 322336.92, 367890.70, 399379.34, 399593.45, 349826.12, 315818.46, 310701.89, 336599.93	9534.35

Using the below mentioned parameters space (parameters and its value) the below mentioned performance scores were achieved.

	Parameters and its value																					
XGBoost Model #	base_score	booster	early_stopping_rounds	min_child_weight	subsample	colsample_bylevel	colsample_bytree	scale_pos_weight	reg_lambda	reg_alpha	learning_rate	max_depth	n_estimators	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
2.1	0.5	gbtree	50								0.01	3	1,000	8,649	10,230	2.438	0.924	8,536	Oct	679	Dec	18,258
2.2	0.5	gbtree	50								0.01		1,000	7,498	8,890	2.108	0.943	7,294	Apr	328	Dec	16,397
3	0.5	gbtree	50								0.01		1,000	8,503	11,192	2.455	0.909	46	May	931	Jan	20,253
4.1	0.5	gbtree	50								0.01	3	1,000	13,261	15,555	3.651	0.825	12,977	Sep	896	Dec	27,025

XGBoost Model 4.2
Cross validation - Five folds.
Features used – month, season, population, income, HDD, CDD, precipitation, max-min-temperature, lag1, lag2, lag3.

For models 4.3 to 4.7 the features used are – population, income, HDD, CDD, precipitation, max-min-temperature, lag1, lag2, lag3. Month, season were removed since these 2 were the lowest important features.

XGBoost Model 4.3 - Cross validation - 5 folds.

XGBoost Model 4.4 - Cross validation – 2 folds.

XGBoost Model 4.5 - Cross validation - 3 folds.

XGBoost Model 4.6 - Cross validation - 12 folds.

XGBoost Model 4.7 - Cross validation - 10 folds.

	Parameters and its value																					
XGBoost Model #	base_score	booster	early_stopping_rounds	min_child_weight	subsample	colsample_bylevel	colsample_bytree	scale_pos_weight	reg_lambda	reg_alpha	learning_rate	max_depth	n_estimators	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
2.1	0.5	gbtree	50								0.01	3	1,000	8,649	10,230	2.438	0.924	8,536	Oct	679	Dec	18,258
2.2	0.5	gbtree	50								0.01		1,000	7,498	8,890	2.108	0.943	7,294	Apr	328	Dec	16,397
3	0.5	gbtree	50								0.01		1,000	8,503	11,192	2.455	0.909	46	May	931	Jan	20,253
4.1	0.5	gbtree	50								0.01	3	1,000	13,261	15,555	3.651	0.825	12,977	Sep	896	Dec	27,025
4.2	0.5	gbtree	50								0.01	3	1,000	13,168	15,185	3.648	0.833	12,826	Sep	396	Dec	26,999
4.3	0.5	gbtree	50								0.01	3	1,000	13,060	15,050	3.620	0.836	12,628	Sep	684	Dec	27,314
4.4	0.5	gbtree	50								0.01	3	1,000	12,412	14,682	3.477	0.844	11,457	Oct	2,367	Jan	23,215
4.5	0.5	gbtree	50								0.01	3	1,000	14,003	16,301	3.884	0.807	13,736	Sep	1,226	Dec	26,466
4.6	0.5	gbtree	50								0.01	3	1,000	14,592	16,662	4.012	0.799	14,592	Oct	1,819	Jul	26,690
4.7	0.5	gbtree	50								0.01	3	1,000	14,331	16,738	3.926	0.797	14,331	Oct	786	Dec	27,274

Inference

- Cross validation does not help increase the score quite well, we will stop using cross validation / folds.

Next steps – tune XGBoost hyper parameters and with the tuned parameters do the prediction.

11.5. XGBoost model 5.1 to 5.6 - Hyperparameters tuning

Features used during hyper tuning are

population, income, HDD, CDD, precipitation, max-min temperature, month, season

Point to note - While doing hyperparameter tuning for all the parameters of XGBoost (as mentioned below) it took over 2 hours without giving any results.

Hence, the following approach was applied.

1) Hyper tuning was done with few parameters in multiple stages.
2) For example, in stage 1 we used learning rate, max depth and number of estimators as the initial set of parameters.
3) For these parameters we defined the parameter space.
4) Use the RandomizedSearchCV with 50 iterations and 'mean squared error' as the scoring parameter to arrive at the best tuned values.
5) With these tuned values we did the prediction, came up with the performance metrics.
6) For the next stage using these tuned values as a baseline, we build up a new model.
7) Picked up the next set of parameters and defined the parameter space.
8) Go to step 4.

	Parameters and its value																					
XGBoost Model #	base_score	booster	early_stopping_rounds	min_child_weight	subsample	colsample_bylevel	colsample_bytree	scale_pos_weight	reg_lambda	reg_alpha	learning_rate	max_depth	n_estimators	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
2.1	0.5	gbtree	50								0.01	3	1,000	8,649	10,230	2.438	0.924	8,536	Oct	679	Dec	18,258
2.2	0.5	gbtree	50								0.01		1,000	7,498	8,890	2.108	0.943	7,294	Apr	328	Dec	16,397
3	0.5	gbtree	50								0.01		1,000	8,503	11,192	2.455	0.909 ✓	46	May	931	Jan	20,253
4.1	0.5	gbtree	50								0.01	3	1,000	13,261	15,555	3.651	0.825	12,977	Sep	896	Dec	27,025
4.2	0.5	gbtree	50								0.01	3	1,000	13,168	15,185	3.648	0.833	12,826	Sep	396	Dec	26,999
4.3	0.5	gbtree	50								0.01	3	1,000	13,060	15,050	3.620	0.836	12,628	Sep	684	Dec	27,314
4.4	0.5	gbtree	50								0.01	3	1,000	12,412	14,682	3.477	0.844	11,457	Oct	2,367	Jan	23,215
4.5	0.5	gbtree	50								0.01	3	1,000	14,003	16,301	3.884	0.807	13,736	Sep	1,226	Dec	26,466
4.6	0.5	gbtree	50								0.01	3	1,000	14,592	16,662	4.012	0.799	14,592	Oct	1,819	Jul	26,690
4.7	0.5	gbtree	50								0.01	3	1,000	14,331	16,738	3.926	0.797	14,331	Oct	786	Dec	27,274
5.1											0.01	3	1,000	5,106	6,166	1.468	0.972 ✓	844	Jun	785	Dec	13,771

Inference

- Positive mfe shows over fitting for the 5.1 XGBoost model but this is second best. With the best 3 features we have had the least amount of over-fitting.
- r2 b/w 0 to 1. Higher r2 shows better prediction performance. This model is the best.
- Lower the mae, rmse, mape values better the prediction performance. This model showed the lowest best scores for all the 3 metrics.
- Best and worst month prediction is Jun, Dec respectively. The highest scores are best but the lowest score is worse.
- XGBoost model 5.1 is a winner so far.

XGBoost model 5.1 to 5.2

Stage No	Parameters and its space	Best hyperparameter values
1	'learning_rate': [0.01, 0.1, 0.2, 0.3], 'max_depth': [3, 4, 5, 6, 7], 'n_estimators': [50, 100, 150, 200],	'learning_rate': 0.2, 'max_depth': 3, 'n_estimators': 200

XGBoost Model 5.2 to 5.3

Stage 2: use the LR, max depth, # estimators as a baseline. Build the model with these parameters and its values. Tune the parameters min child weight and subsample.

Stage No	Parameters and its space	Best hyperparameter values	Time taken (seconds)
1	'learning_rate': [0.01, 0.1, 0.2, 0.3], 'max_depth': [3, 4, 5, 6, 7], 'n_estimators': [50, 100, 150, 200],	'learning_rate': 0.2, 'max_depth': 3, 'n_estimators': 200	43.89
2	'min_child_weight': [1, 5, 10], 'subsample': [0.6, 0.7, 0.8, 0.9, 1.0]	'min_child_weight': 1, 'subsample': 0.6	11.98

XGBoost Model 5.3 to 5.4

Stage 3: use the LR, depth, estimators, min child weight and subsample as a baseline. Build the model with these parameters and its values. Tune the parameters 'colsample_bytree' and 'colsample_bylevel'

XGBoost Model 5.4 to 5.5

Stage 4: use the LR, depth, estimators, min child weight, subsample, 'colsample_bytree' and 'colsample_bylevel' as a baseline. Build the model with these parameters and its values. Tune the parameters 'scale_pos_weight'.

XGBoost Model 5.5 to 5.6

Stage 5: use the LR, depth, estimators, min child weight, subsample, 'colsample_bytree', 'colsample_bylevel' and 'scale_pos_weight' as a baseline. Build the model with these parameters and its values. Tune the parameters 'reg_alpha', 'reg_lambda'

Inference

- Adding 'reg_alpha': 1.0, 'reg_lambda': 10.0 (the hyper tuned value) gives slight improvement but not as best as XGBoost models 5.1. Hence, we ignore them.

	Parameters and its value																						
XGBoost Model #	base_score	booster	early_stopping_rounds	min_child_weight	subsample	colsample_bylevel	colsample_bytree	scale_pos_weight	reg_lambda	reg_alpha	learning_rate	max_depth	n_estimators	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
2.1	0.5	gbtree	50								0.01	3	1,000	8,649	10,230	2.438	0.924	8,536	Oct	679	Dec	18,258
2.2	0.5	gbtree	50								0.01		1,000	7,498	8,890	2.108	0.943	7,294	Apr	328	Dec	16,397
3	0.5	gbtree	50								0.01		1,000	8,503	11,192	2.455	0.909 ✓	46	May	931	Jan	20,253
4.1	0.5	gbtree	50								0.01	3	1,000	13,261	15,555	3.651	0.825	12,977	Sep	896	Dec	27,025
4.2	0.5	gbtree	50								0.01	3	1,000	13,168	15,185	3.648	0.833	12,826	Sep	396	Dec	26,999
4.3	0.5	gbtree	50								0.01	3	1,000	13,060	15,050	3.620	0.836	12,628	Sep	684	Dec	27,314
4.4	0.5	gbtree	50								0.01	3	1,000	12,412	14,682	3.477	0.844	11,457	Oct	2,367	Jan	23,215
4.5	0.5	gbtree	50								0.01	3	1,000	14,003	16,301	3.884	0.807	13,736	Sep	1,226	Dec	26,466
4.6	0.5	gbtree	50								0.01	3	1,000	14,592	16,662	4.012	0.799	14,592	Oct	1,819	Jul	26,690
4.7	0.5	gbtree	50								0.01	3	1,000	14,331	16,738	3.926	0.797	14,331	Oct	786	Dec	27,274
5.1											0.01	3	1,000	5,106	6,166	1.468	0.972 ✓	844	Jun	785	Dec	13,771
5.2											0.2	3	200	5,986	7,309	1.717	0.961	-4,823	May ✓	183	Oct	12,458
5.3				1	0.6						0.2	3	200	6,090	7,197	1.744	0.962	2,838	Mar	1,223	Jan	13,397
5.4				1	0.6	0.8	0.9				0.2	3	200	4,586	5,367	1.342	0.979 ✓	398	May	480	Apr ✓	8,911
5.5				1	0.6	0.8	0.9	1			0.2	3	200	4,586	5,367	1.342	0.979 ✓	398	May	480	Apr ✓	8,911
5.6				1	0.6	0.8	0.9	1	10	10	0.2	3	200	7,128	8,238	2.000	0.951	4,152	Mar	672	Dec	15,380

Stage No	Parameters and its space	Best hyperparameter values	Time taken (seconds)
1	'learning_rate': [0.01, 0.1, 0.2, 0.3], 'max_depth': [3, 4, 5, 6, 7], 'n_estimators': [50, 100, 150, 200],	'learning_rate': 0.2, 'max_depth': 3, 'n_estimators': 200	43.89
2	'min_child_weight': [1, 5, 10], 'subsample': [0.6, 0.7, 0.8, 0.9, 1.0]	'min_child_weight': 1, 'subsample': 0.6	11.98
3	'colsample_bytree': [0.6, 0.7, 0.8, 0.9, 1.0], 'colsample_bylevel': [0.6, 0.7, 0.8, 0.9, 1.0]	'colsample_bylevel': 0.8, 'colsample_bytree': 0.9	14.63
4	'scale_pos_weight': [1, 2, 3, 4]	'scale_pos_weight': 1	2.48
5	'reg_lambda': [0.01, 0.1, 1.0, 10.0, 100.0], 'reg_alpha': [0, 0.1, 1.0, 10.0]	'reg_alpha': 1.0, 'reg_lambda': 10.0	13.26

Inference

- Post the various stages i.e., Stage 1 to Stage 5 the various parameters that were tuned and their tuned values are reflected in the table above.
- No values were specified for base score, booster and early stopping rounds for all these stages.
- We see that by splitting the parameter space it took less than 90 seconds to tune all the parameters. If we had tuned all the parameters in one go with the above set of values in the parameter space, it took over 2 hours without producing any results.
- Model 5.1 – The parameter space used was LR = 0.01, max depth = 3, # estimators = 1000
- Stage 1 (model 5.1 to model 5.2) -> When we tuned the XGBoost model 5.1 we got 5.2 in which the '# estimators' decreased from 1,000 to 200, 'max depth' value remained the same, LR changed from 0.01 to 0.2. The revised values for # estimators and LR were used for the remaining XGBoost models until model 5.6.
- Stage 2 (model 5.2 to model 5.3) -> With the tuned values of 'min child weight' and 'sub sample' there were no improvement in the scores for model 5.3.
- Stage 3 (model 5.3 to model 5.4) -> With the tuned values of 'col sample by tree' and 'col sample by level' there were significant improvement in the performance scores in model 5.4.
- Stage 4 (model 5.4 to model 5.5) -> No change in the performance scores for model 5.4 and model 5.5. This implies that whether we explicitly specify 'scale_pos_weight' = 1 it does not matter.
- Stage 5 (model 5.5 to model 5.6) -> With the tuned values of 'reg_lambda' and 'reg_alpha' the performance scores degraded in model 5.6.

11.6. XGBoost model 5.7, 5.8, 5.9 - Hyperparameters tuning, exploring the boundaries

Next steps of hyper tuning

We will ignore scale_pos_weight as with and without it the performance scores are same. The parameter space used for the above set of parameters were

- 'min_child_weight': [1, 5, 10] -> 1 is OK as it cannot take decimal values.
- 'subsample': [0.6, 0.7, 0.8, 0.9, 1.0] -> 0.6, Explore with lower values.
- 'colsample_bytree': [0.6, 0.7, 0.8, 0.9, 1.0] -> 0.9 is OK.
- 'colsample_bylevel': [0.6, 0.7, 0.8, 0.9, 1.0] -> 0.8 is OK.
- 'learning_rate': [0.01, 0.1, 0.2, 0.3] -> 0.2 is OK.
- 'max_depth': [3, 4, 5, 6, 7] -> 3, explore with lower values.
- 'n_estimators': [50, 100, 150, 200] -> 200, Explore with higher values.

Hence, we have the new parameter space defined as

- 'min_child_weight' = 1
- 'colsample_bytree' = 0.9
- 'colsample_bylevel '= 0.8
- 'learning_rate' = 0.2
- 'subsample': [0.4, 0.5, 0.6]
- 'max_depth': [1, 2, 3]
- 'n_estimators': [200, 500, 1,000]

After doing the fine tuning the refined parameters are

- 'min_child_weight' = 1
- 'colsample_bytree'= 0.9
- 'colsample_bylevel'= 0.8
- 'learning_rate'= 0.2
- 'subsample' = 0.6
- 'max_depth' = 3
- 'n_estimators' = 1,000

With the above tuned parameters, we produce an XGBoost model 5.7 and do a prediction.

In model 5.7 we add base_score = 0.5, booster = gbtree and early_stopping_rounds=50 to get model 5.8.

All these models had the features 'population', 'income', 'HDD', 'CDD', 'precipitation', 'max_min_temp'.
With the same hyper tuned values, we run a model 5.9 with just lag 1, lag 2, lag 3

Trials	Inference / Results
Various values of 'early stopping round' = 10, 50, 25, 100, 1000	If we use 'early stopping round' = 50, 25, 100, 1000 the performance metrics are same. For value = 10 the performance starts degrading hence we will stick with 50
Explicitly specifying the base_score value	With or without explicit mentioning base_score does not affect the outcome of the performance metrics (rmse etc.).
Different values of booster = gbtree, gblinear, dart	With booster = gblinear, the performance scores are worse so ignoring it. With booster = dart, the performance scores are same as gbtree.

The final stats for XGBoost are as shown below.

XGBoost Model #	Parameters and its value													mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
	base_score	booster	early_stopping_rounds	min_child_weight	subsample	colsample_bylevel	colsample_bytree	scale_pos_weight	reg_lambda	reg_alpha	learning_rate	max_depth	n_estimators									
2.1	0.5	gbtree	50								0.01	3	1,000	8,649	10,230	2.438	0.924	8,536	Oct	679	Dec	18,258
2.2	0.5	gbtree	50								0.01		1,000	7,498	8,890	2.108	0.943	7,294	Apr	328	Dec	16,397
3	0.5	gbtree	50								0.01		1,000	8,503	11,192	2.455	0.909 ✓	46	May	931	Jan	20,253
4.1	0.5	gbtree	50								0.01	3	1,000	13,261	15,555	3.651	0.825	12,977	Sep	896	Dec	27,025
4.2	0.5	gbtree	50								0.01	3	1,000	13,168	15,185	3.648	0.833	12,826	Sep	396	Dec	26,999
4.3	0.5	gbtree	50								0.01	3	1,000	13,060	15,050	3.620	0.836	12,628	Sep	684	Dec	27,314
4.4	0.5	gbtree	50								0.01	3	1,000	12,412	14,682	3.477	0.844	11,457	Oct	2,367	Jan	23,215
4.5	0.5	gbtree	50								0.01	3	1,000	14,003	16,301	3.884	0.807	13,736	Sep	1,226	Dec	26,466
4.6	0.5	gbtree	50								0.01	3	1,000	14,592	16,662	4.012	0.799	14,592	Oct	1,819	Jul	26,690
4.7	0.5	gbtree	50								0.01	3	1,000	14,331	16,738	3.926	0.797	14,331	Oct	786	Dec	27,274
5.1	0.5	gbtree	50								0.01	3	1,000	5,106	6,166	1.468	0.972 ✓	844	Jun	785	Dec	13,771
5.2											0.2	3	200	5,986	7,309	1.717	0.961	-4,823	May ✓	183	Oct	12,458
5.3				1	0.6						0.2	3	200	6,090	7,197	1.744	0.962	2,838	Mar	1,223	Jan	13,397
5.4				1	0.6	0.8	0.9				0.2	3	200	4,586	5,367	1.342	0.979 ✓	398	May	480	Apr ✓	8,911
5.5				1	0.6	0.8	0.9	1			0.2	3	200	4,586	5,367	1.342	0.979 ✓	398	May	480	Apr ✓	8,911
5.6				1	0.6	0.8	0.9	1	10	10	0.2	3	200	7,128	8,238	2.000	0.951	4,152	Mar	672	Dec	15,380
5.7				1	0.6	0.8	0.9	1			0.2	3	1,000 ✓	4,094 ✓	4,858 ✓	1.185 ✓	0.983 ✓	905	Mar ✓	30	Apr ✓	7,415
5.8	0.5	gbtree		1	0.6	0.8	0.9	1			0.2	3	1,000 ✓	4,401 ✓	5,214 ✓	1.283 ✓	0.980 ✓	695	Mar	858	Dec ✓	8,907
5.9	0.5	gbtree	50								0.1	3	10,000	6,239	7,631	1.772	0.958	5,931	Oct	851	Dec	13,671

Table 3: Performance of the various XGBoost models

Inference

- The best score is for model 5.7 (highlighted in green border) which has the lowest mae, rmse, mape scores amongst all the XGBoost models. The R2 score (= 0.983) is very close to 1.
- The mfe is positive which shows over fitting. But the score is comparatively low 905.
- The best month prediction score (=30) for Mar, has the lowest deviation across all the models.
- The worst month prediction score (=7,415) for Apr, is also the lowest amongst all the models.

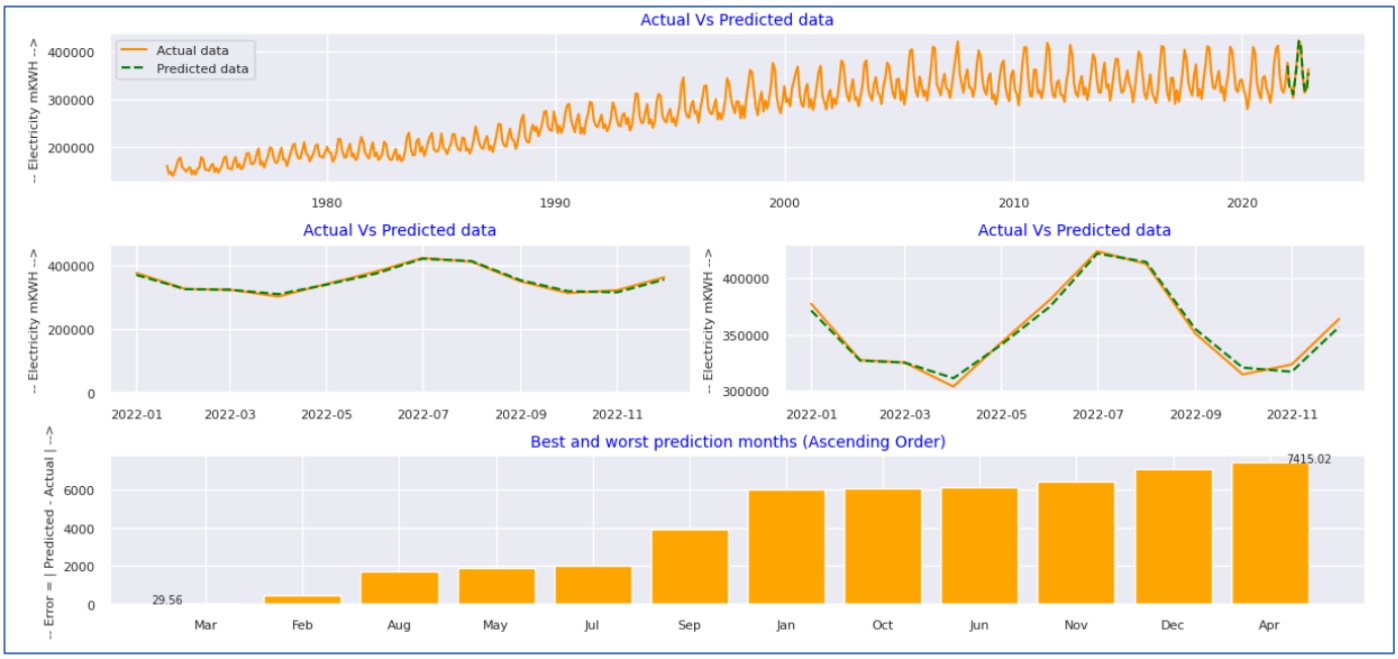


Image 29: Stats for XGBoost model 5.7

The performance so far for all the models are as shown below.

Performance of the various models										
Sl No	Model details	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
1	Auto regression with lags (1,2, 3 years) with linear regression	9,678	12,762	2.881	0.885	-1,520	Feb	1,633	Jan	19,367
2	Auto regression with lags (1,2, 3 years) with random forest	38,518	13,568	11.117	0.870	-12	Feb	1,793	Jul	28,979
3	Auto regression with lags (1,2, 3 years) with random forest & decomposed data	7,753	9,706	2.213	0.928	3,489	Oct	✓ 414	Aug	17,106
4	Auto regression with lags (1,2, 3 years) with linear regression & decomposed data	9,350	11,946	2.779	0.891	-1,302	Apr	✓ 62	Jun	19,542
5a	ARIMA model (10, 2, 1)	12,006	14,568	3.258	0.846	9,096	Sep	773	Jan	23,868
5b	ARIMA model (09, 2, 1)	13,690	16,697	3.706	0.798	6,677	Mar	3,422	Jul	33,115
5c	ARIMA model (10, 1, 1)	13,763	16,293	3.763	0.807	10,759	Nov	798	Jan	26,713
6a	SARIMA model (0, 1, 1, 2, 0, 0, 12)	41,842	17,224	11.515	0.785	14,347	Feb	3,577	Jan	30,957
6b	SARIMA model (1, 1, 1, 2, 0, 0, 12)	39,627	16,208	10.904	0.809	11,968	Feb	1,062	Jan	33,432
6c	SARIMA model (0, 1, 1, 2, 1, 2, 12)	41,186	14,262	11.374	0.853	11,621	Sep	✓ 638	Dec	24,055
7.1	SARIMAX model, 3 exogenous features	39,774	9,320	11.088	✓ 0.937	4,678	Mar	✓ 322	Dec	15,648
7.2	SARIMAX phase 15: 6 exogeneous features	41,235	6,338	11.535	✓ 0.971	✓ 2,599	Jul	820	Dec	✓ 12,297
8	XGBoost model 5.7	✓ 4,094	✓ 4,858	✓ 1.185	✓ 0.983	✓ 905	Mar	✓ 30	Apr	✓ 7,415
	Tick marks is	x < = 5%	x < = 5%	x < = 5%	x > = 75%	15%<x< 30%		x < = 20%		x < = 20%

Inference

- XGBoost ticks all the boxes and is the best model so far.

12. Chapter 6: LSTM (Long Short-Term Memory)

12.1. Strategy for LSTM

LSTM (Long Short-Term Memory) model is used in DL (Deep Learning) and ML (Machine Learning). It has a specialized form of RNN (Recurrent Neural Network) designed to capture and model, long-term dependencies and patterns in sequential data. It is well-suited for tasks involving time series data, NLP (Natural Language Processing), speech recognition etc.

LSTM networks is used to address the vanishing gradient problem which is often faced by traditional RNNs. Traditional RNN has a limited capability to capture long-term dependencies in data. LSTM uses a complex structure of recurrent units called "cells" that maintain a memory state, allowing them to learn and store information over longer sequences.

There are 600 data points. Monthly data from Jan1973 until Dec2022 for 50 years (50 years x 12 months = 600).
Train data is from Jan1973 until Dec2021. 588 data points (49 years x 12 months = 588).
Test data is from Jan2022 until Dec2022. 12 data points (01 year x 12 months = 12).

```
df_ltsm_train = df_ltsm[:-12] #data from Jan1973 until Dec2021
df_ltsm_test = df_ltsm[-12:] #data from Jan to Dec 2022
```

Before we go any further, we normalize the tsd between 0 to 1. This is to preempt the model to get confused since the tsd varies from 160217.989 (160k) until 363624.583 (320k)

```
ltsm_scaler = MinMaxScaler() #convert the tsd in a scale of 0 to 1, avoid the model to get confused
ltsm_scaler.fit(df_ltsm_train)
na_lscaled_train = ltsm_scaler.transform(df_ltsm_train) #data type numpy.ndarray
na_lscaled_test = ltsm_scaler.transform(df_ltsm_test) #data type numpy.ndarray
```

```
na_lscaled_train.shape, df_ltsm_train.shape, df_ltsm_test.shape, na_lscaled_test.shape
#there are 600 data points from 1973 until 2022
#2022-1973 = len(df_ltsm) = 600
#when divided fir train Vs test we get 600-12 = 588 Vs 12 data points
```

```
((588, 1), (588, 1), (12, 1), (12, 1))
```

We then use a generator to process and feed sequential data into the LSTM model. This is a monthly tsd. Hence, we will be using 12 months data as an input to predict the data for the 13th month.

```
# generator definition
nof_input = 12 #use 12months data to predict data for the 13th month
nof_features = 1 # we are using only 1 feature to do ts prediction and hence n_features = 1
generator = TimeseriesGenerator(na_lscaled_train, na_lscaled_train, length=nof_input, batch_size=1)
```

The number of samples we then have is

```
print('Number of samples: %d' % len(generator)) # number of samples, 576 = 588(# of train data) - 12(# of i/p)
# #print each sample
# for i in range(len(generator)):
#  x, y = generator[i]
#  print('%s => %s' % (x, y))
```

```
Number of samples: 576
```

The output value generated in generator [0] i.e., 0.06365979 becomes the 12th input value in generator [1]

```
1 X,y = generator[0]
2 print(f'Given the Array: -> {X.flatten()}')
3 print(f'Predict this y, this is the actual y: -> {y}')
```

```
Given the Array: -> [0.07309717 0.01399418 0.03036396 0.         0.02765941 0.07673168
 0.12098757 0.13385882 0.06125142 0.05176347 0.0302904  0.04966357]
Predict this y, this is the actual y: -> [[0.06365979]]
```

```
[281] 1 X,y = generator[1]
      2 print(f'Given the Array: -> {X.flatten()}')
      3 print(f'Predict this y, this is the actual y: -> {y}')
```

```
Given the Array: -> [0.01399418 0.03036396 0.         0.02765941 0.07673168 0.12098757
 0.13385882 0.06125142 0.05176347 0.0302904  0.04966357 0.06365979]
Predict this y, this is the actual y: -> [[0.01119397]]
```

The output value generated in generator [0] i.e., 0.06365979 and generator [1] i.e., 0.01119397 becomes the 11th and 12th input values in generator [2]

```
1 X,y = generator[0]
2 print(f'Given the Array: -> {X.flatten()}')
3 print(f'Predict this y, this is the actual y: -> {y}')
```

```
Given the Array: -> [0.07309717 0.01399418 0.03036396 0.         0.02765941 0.07673168
 0.12098757 0.13385882 0.06125142 0.05176347 0.0302904  0.04966357]
Predict this y, this is the actual y: -> [[0.06365979]]
```

```
] 1 X,y = generator[1]
  2 print(f'Given the Array: -> {X.flatten()}')
  3 print(f'Predict this y, this is the actual y: -> {y}')
```

```
Given the Array: -> [0.01399418 0.03036396 0.         0.02765941 0.07673168 0.12098757
 0.13385882 0.06125142 0.05176347 0.0302904  0.04966357 0.06365979]
Predict this y, this is the actual y: -> [[0.01119397]]
```

```
] 1 X,y = generator[2]
  2 print(f'Given the Array: -> {X.flatten()}')
  3 print(f'Predict this y, this is the actual y: -> {y}')
```

```
Given the Array: -> [0.03036396 0.         0.02765941 0.07673168 0.12098757 0.13385882
 0.06125142 0.05176347 0.0302904  0.04966357 0.06365979 0.01119397]
Predict this y, this is the actual y: -> [[0.03810238]]
```

This goes on and on, to prepare the data which will be then fed into the LSTM model.

In our case there are 576 samples of input and output generated by the generator which will be used as an input to the LSTM model. Each of these 576 batches has 12 input and only 1 output.

In short for input size = 3, the data that will be fed by the generator in the LSTM model will be:
[1, 2, 3] -> [4], then
[2, 3, 4] -> [5], then
[3, 4, 5] -> [6], etc.

12.2. LSTM model 1, 2, 3, 4 – The first stab

LSTM model 1 specifications

Input layer (IL) - 12
Hidden layer (HL) - 100, A/F = ReLU
Output layer (OL) - 1
Optimizer = adam
Loss = mse (mean squared error)
# epochs = 50

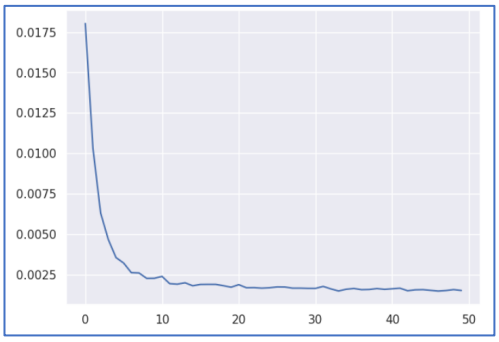


Image 30: Loss per epoch Vs Number of epoch

After 20 epochs the changes in loss is still < 0.0025. Hence, we can stop after 20 epochs and not go until 50.

We change various parameters for the below mentioned 4 models. Inference is as mentioned below.

Performance of the LSTM models										
Sl No	LSTM model details (# nodes in Input layer, # nodes in hidden layer - Activation function,#nodes in output layer, optimizer, loss function, # epochs)	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
1	LSTM model 1 (12, 100-relu, 1, adam, mse, 50)	12,567	15,744	3.482	0.820	11,180	Feb	845	Dec	29,295
2	LSTM model 2 (12, 100-relu, 1, adam, mse, 25)	10,788	13,269	2.994	0.972	7,768	Feb	839	Jan	25,950
3	LSTM model 3 (12, 50-relu, 1, adam, mse,25)	11,113	14,426	3.090	0.849	7,695	Feb	843	Jan	29,670
4	LSTM model 4 (6, 50-relu, 1, adam, mse,25)	39,686	43,016	11.414	-0.076	-19,815				

In model 2 # epochs are halved from 50 to 25
Inference - Slight improvement in the performance metrics. Hence, stick with 25 epochs.

In model 3 # neurons in the HL are halved from 100 to 50
Inference - Slight degradation in the performance metrics than the 2nd LSTM model but better than the first.

In model 4 # input features are halved from 12 to 6
Inference - Poor performance metrics than all the 3 LSTM models. Ignore this model. Stick with the number of input features as 12

12.3. LSTM model 5, 6 – Hyperparameter tuning – RS, BO

We are going to use a Keras model to predict the hyperparameters of the LSTM model. To begin with we will be considering only 1 hidden layer (HL). We need to predict the # nodes in this HL.

- Hidden layer activation function - we will be using ReLU or Tanh activation function in the hidden layer
  - ReLU (Rectified Linear Unit) introduces non-linearity into the model, helping capture complex patterns in the data.
  - Tanh (Hyperbolic Tangent) squashes input values to the range [-1, 1], which can help in capturing both positive and negative patterns in the data.

- Output layer activation function
  - Linear Activation (for Regression): Here we are performing a regression task where we want to predict a numerical value (i.e., electricity generation). We should use a linear activation function in the output layer. Linear activation allows the model to produce a continuous range of values.

- Loss function - We will use 'mse' (mean squared error) as the loss function
- Metric - We will use 'mae', 'rmse' as the metrics
- Tuner – Random Search, Bayesian Optimization

Tuner name	**Random Search**	**Bayesian Optimization**
Pros	It is a straightforward and an efficient method for hyperparameter tuning. It randomly samples hyperparameters from specified ranges, making it relatively fast and easy to set up.	It is more sophisticated and adaptive compared to random search. It models the surrogate function to intelligently explore the hyperparameter space. It often converges faster to optimal hyperparameters.
Cons	It does not adapt its search strategy based on the results of the previous trials. It might require a larger number of trials to find the best hyperparameters compared to Bayesian optimization.	It may require more computational resources compared to random search. It is especially useful when we have a limited budget of trials or want to optimize resource-intensive models.

Tuner definition	`tuner_rs = RandomSearch(`
	`  build_model,`
	`  objective='val_loss',  # Speci`
	`  max_trials=50,  # Number of hy`
	`  directory='my_dir_rs',  # Dire`
	`  project_name='my_project_rs'`
	`)`
		`tuner_bo = BayesianOptimization(`
		`  build_model,  # Your`
		`  objective='val_mean_squared_er`
		`  max_trials=50,  # Numbe`
		`  num_initial_points=5,  # Numbe`
		`  directory='my_dir_bo',  # Di`
		`  project_name='my_project_bo'`
		`)`

Early stopping definition	`# Create the EarlyStopping callback`
	`early_stopping = EarlyStopping(`
	`  monitor='val_loss',  # Monitor`
	`  patience=10,  # Number`
	`  restore_best_weights=True  # Rest`
	`)`
		`# Create the EarlyStopping callback`
		`early_stopping = EarlyStopping(`
		`  monitor='val_loss',  # Monitor`
		`  patience=10,  # Number`
		`  restore_best_weights=True  # Rest`
		`)`

# nodes in the HL	192 for AF = ReLU in the 1 HL	256 for AF = ReLU in the 1 HL

Table 4: Tuners - Radom search Vs Bayesian optimization. Pros and Cons.

Performance of the LSTM models										
Sl No	LSTM model details (# nodes in Input layer, # nodes in hidden layer - Activation function,#nodes in output layer, optimizer, loss function, # epochs)	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
1	LSTM model 1 (12, 100-relu, 1, adam, mse, 50)	12,567	15,744	3.482	0.820	11,180	Feb	845	Dec	29,295
2	LSTM model 2 (12, 100-relu, 1, adam, mse, 25)	10,788	13,269	2.994	0.972	7,768	Feb	839	Jan	25,950
3	LSTM model 3 (12, 50-relu, 1, adam, mse,25)	11,113	14,426	3.090	0.849	7,695	Feb	843	Jan	29,670
4	LSTM model 4 (6, 50-relu, 1, adam, mse,25)	39,686	43,016	11.414	-0.076	-19,815				
5	LSTM model 5 (12, 1HL-relu, [192], 1, adam, mse, 100), Tuner-RS	10,773	13,104	3.055	0.875	9,781	Oct	1,934	Jan	25,157
6	LSTM model 6 (12, 1HL-relu, [256], 1, adam, mse, 100), Tuner-BO	10,794	13,365	3.049	0.870	9,852	Sep	526	Jan	27,546

After using RS tuner, BO tuner the number of nodes in the hidden layer is 192, 256 respectively. Not very good improvement in any performance scores.

12.4. LSTM model 7, 8 – More hyperparameter tuning

Use tanh as an AF in the hidden layer instead of ReLU

Performance of the LSTM models										
Sl No	LSTM model details (# nodes in Input layer, # nodes in hidden layer - Activation function,#nodes in output layer, optimizer, loss function, # epochs)	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
1	LSTM model 1 (12, 100-relu, 1, adam, mse, 50)	12,567	15,744	3.482	0.820	11,180	Feb	845	Dec	29,295
2	LSTM model 2 (12, 100-relu, 1, adam, mse, 25)	10,788	13,269	2.994	0.972	7,768	Feb	839	Jan	25,950
3	LSTM model 3 (12, 50-relu, 1, adam, mse,25)	11,113	14,426	3.090	0.849	7,695	Feb	843	Jan	29,670
4	LSTM model 4 (6, 50-relu, 1, adam, mse,25)	39,686	43,016	11.414	-0.076	-19,815				
5	LSTM model 5 (12, 1HL-relu, [192], 1, adam, mse, 100), Tuner-RS	8,644	10,440	2.436	0.921	5,808	Jun	1,512	Jan	21,081
6	LSTM model 6 (12, 1HL-relu, [256], 1, adam, mse, 100), Tuner-BO	10,732	13,529	3.015	0.867	10,128	Sep	497	Jan	29,157
7	LSTM model 7 (12, 1HL-tanh, [256], 1, adam, mse, 100), Tuner-RS	13,565	16,715	3.805	0.797	12,293	Aug	1,027	Jan	28,792
8	LSTM model 8 (12, 1HL-tanh, [256], 1, adam, mse, 100), Tuner-BO	10,564	12,538	2.959	0.886	5,209	Jun	1,019	Jan	21,697

Inference – When using 1 hidden layer, and finding the best parameter values, we get 192 nodes in the hidden layer for Tuner – RS which gives us overall the best performance scores so far for mae, rmse, map, R2. mfe is the 2nd best score.

In either case (i.e., using Random Search or the Bayesian Optimization tuners) the Activation Function = ReLU gives the best performance scores. Hence, we will stop using Activation Function = tanh going forward.

12.5. LSTM model 9, 10, 11, 12 – Multiple hidden layers

Point to note is that for the same set of hyperparameters and its values the LSTM model gives different values for the performance metric like rmse, mae etc. every time we run the same model. This is because of various factors. Some of them being.

i. Random initialization: LSTM models, like other neural networks, typically involve random initialization of weights and biases. This random initialization introduces variability in the initial state of the network. Therefore, even for the same dataset, running the training process multiple times with the same hyperparameters can lead to slightly different initial conditions. This can result in variations in model performance.

ii. Stochastic optimization: Training DL models like LSTM often involves stochastic optimization algorithms like Stochastic Gradient Descent (SGD) or variants of it. These algorithms rely on randomness, such as selecting mini-batches of data, and the order of data samples can vary. Consequently, the training process may follow slightly different convergence paths in different runs, leading to variations in RMSE.

iii. Hyperparameter initialization: Some hyperparameters (e.g., learning rates, dropout rates) may be initialized with random values, which can lead to different model behavior and performance.

To address this variability, as much as possible we have set random seeds. To ensure reproducibility, we have set random seeds for libraries like NumPy and TensorFlow at the beginning of our code to control randomness in weight initialization, data shuffling, and other random operations. "np.random.seed(0)"

For the models 9 to 12 we have used the following architecture, model, tuner

Model 9 – 2 hidden layers each having the ReLU AF, Tuner – Random Search
Model 10 – 2 hidden layers each having the ReLU AF, Tuner – Bayesian Optimization
Model 11 – 3 hidden layers each having the ReLU AF, Tuner – Random Search
Model 12 - 3 hidden layers each having the ReLU AF, Tuner – Bayesian Optimization

Performance of the LSTM models										
Sl No	LSTM model details (# nodes in Input layer, # nodes in hidden layer - Activation function,#nodes in output layer, optimizer, loss function, # epochs)	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
1	LSTM model 1 (12, 100-relu, 1, adam, mse, 50)	12,567	15,744	3.482	0.820	11,180	Feb	845	Dec	29,295
2	LSTM model 2 (12, 100-relu, 1, adam, mse, 25)	10,788	13,269	2.994	0.972	7,768	Feb	839	Jan	25,950
3	LSTM model 3 (12, 50-relu, 1, adam, mse,25)	11,113	14,426	3.090	0.849	7,695	Feb	843	Jan	29,670
4	LSTM model 4 (6, 50-relu, 1, adam, mse,25)	39,686	43,016	11.414	-0.076	-19,815				
5	LSTM model 5 (12, 1HL-relu, [192], 1, adam, mse, 100), Tuner-RS	8,644	10,440	2.436	0.921	5,808	Jun	1,512	Jan	21,081
6	LSTM model 6 (12, 1HL-relu, [256], 1, adam, mse, 100), Tuner-BO	10,732	13,529	3.015	0.867	10,128	Sep	497	Jan	29,157
7	LSTM model 7 (12, 1HL-tanh, [256], 1, adam, mse, 100), Tuner-RS	13,565	16,715	3.805	0.797	12,293	Aug	1,027	Jan	28,792
8	LSTM model 8 (12, 1HL-tanh, [256], 1, adam, mse, 100), Tuner-BO	10,564	12,538	2.959	0.886	5,209	Jun	1,019	Jan	21,697
9	LSTM model 9 - 2 HLs [160,224] each having AF = relu, Tuner - RS	10,079	13,420	2.808	0.869	9,723	Feb	474	Jan	29,533
10	LSTM model 10 - 2 HLs [96,256] each having AF = relu, Tuner - BO	11,065	13,536	3.123	0.867	10,478	Oct	1,184	Jan	29,476
11	LSTM model 11 - 3 HLs [32, 224, 160] each having AF = relu, Tuner - RS	11,181	13,480	3.164	0.868	10,342	Aug	172	Jan	25,787
12	LSTM model 12 - 3 HLs [128, 192, 160] each having AF = relu, Tuner - BO	11,232	13,943	3.173	0.859	10,452	Oct	2,179	Jan	29,021

Inference
For all the models using 2 and 3 hidden layers, model 9 so far has the best performance scores.
Considering all the LSTM models so far model 9 has the 2nd best performance score

Inference – do the prediction for 3HL for tanh AF in all the 3 HL

Performance of the LSTM models										
Sl No	LSTM model details (# nodes in Input layer, # nodes in hidden layer - Activation function,#nodes in output layer, optimizer, loss function, # epochs)	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
1	LSTM model 1 (12, 100-relu, 1, adam, mse, 50)	12,567	15,744	3.482	0.820	11,180	Feb	845	Dec	29,295
2	LSTM model 2 (12, 100-relu, 1, adam, mse, 25)	10,788	13,269	2.994	✓ 0.972	7,768	Feb	839	Jan	25,950
3	LSTM model 3 (12, 50-relu, 1, adam, mse,25)	11,113	14,426	3.090	0.849	7,695	Feb	843	Jan	29,670
4	LSTM model 4 (6, 50-relu, 1, adam, mse,25)	39,686	43,016	11.414	-0.076	-19,815				
5	LSTM model 5 (12, 1HL-relu, [192], 1, adam, mse, 100), Tuner-RS	✓ 8,644	✓ 10,440	✓ 2.436	✓ 0.921	✓ 5,808	Jun	1,512	Jan	✓ 21,081
6	LSTM model 6 (12, 1HL-relu, [256], 1, adam, mse, 100), Tuner-BO	10,732	13,529	3.015	0.867	10,128	Sep	497	Jan	29,157
7	LSTM model 7 (12, 1HL-tanh, [256], 1, adam, mse, 100), Tuner-RS	13,565	16,715	3.805	0.797	12,293	Aug	1,027	Jan	28,792
8	LSTM model 8 (12, 1HL-tanh, [256], 1, adam, mse, 100), Tuner-BO	10,564	12,538	2.959	0.886	✓ 5,209	Jun	1,019	Jan	21,697
9	LSTM model 9 - 2 HLs [160,224] each having AF = relu, Tuner - RS	✓ 10,079	13,420	✓ 2.808	0.869	9,723	Feb	474	Jan	29,533
10	LSTM model 10 - 2 HLs [96,256] each having AF = relu, Tuner - BO	11,065	13,536	3.123	0.867	10,478	Oct	1,184	Jan	29,476
11	LSTM model 11 - 3 HLs [32, 224, 160] each having AF = relu, Tuner - RS	11,181	13,480	3.164	0.868	10,342	Aug	✓ 172	Jan	25,787
12	LSTM model 12 - 3 HLs [128, 192, 160] each having AF = relu, Tuner - BO	11,232	13,943	3.173	0.859	10,452	Oct	2,179	Jan	29,021
13	LSTM model 13 - 3 HLs [224, 192, 160] each having AF = tanh, Tuner - RS	10,278	12,078	2.890	0.894	✓ 5,695	Jun	2,296	Jan	✓ 20,868
14	LSTM model 14 - 3 HLs [96, 224, 128] each having AF = tanh, Tuner - BO	11,485	14,265	3.240	0.852	9,760	Sep	541	May	23,448
	Tick marks is	x < = 5%	x < = 5%	x < = 5%	x > = 95%	70%<x< 80%		x < = 5%		x < = 5%

Table 5: Performance of the various LSTM models

Inference –

- Not much improvement in the performance scores using 3 hidden layers and activation function = tanh
- We also tried using the independent features as an input to the model. However, the performance scores were dismal. Hence, we are ignoring them.
- If we look at all the LSTM models studied so far, the best performance score is given by model 5.

One of the reasons why we did not get very good performance scores for the LSTM models may be because of the length of the test data range when compared to the length of the train data range.
We had test data from 1973 until 2021 = 49 years x 12 months = 588 data points and train data range 1 year x 12 months = 12 data points.

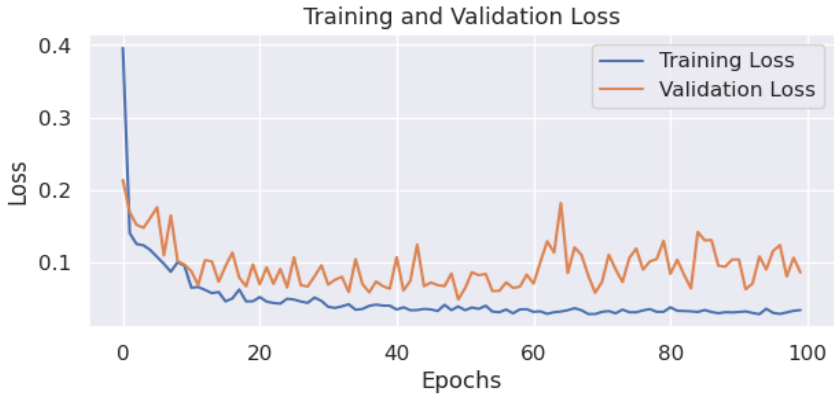


Image 31: Training & validation loss

12.6. Chapter 7: Conclusion and Future work

Performance of the various models										
Sl No	Model details	mae	rmse	mape	R2	mfe	Best month	Best month value	Worst month	Worst month value
1	Auto regression with lags (1,2, 3 years) with linear regression	9,678	12,762	2.881	0.885	-1,520	Feb	1,633	Jan	19,367
2	Auto regression with lags (1,2, 3 years) with random forest	38,518	13,568	11.117	0.870	-12	Feb	1,793	Jul	28,979
3	Auto regression with lags (1,2, 3 years) with random forest & decomposed data	7,753	9,706	2.213	0.928	3,489	Oct	414	Aug	17,106
4	Auto regression with lags (1,2, 3 years) with linear regression & decomposed data	9,350	11,946	2.779	0.891	-1,302	Apr	✓ 62	Jun	19,542
5a	ARIMA model (10, 2, 1)	12,006	14,568	3.258	0.846	9,096	Sep	773	Jan	23,868
5b	ARIMA model (09, 2, 1)	13,690	16,697	3.706	0.798	6,677	Mar	3,422	Jul	33,115
5c	ARIMA model (10, 1, 1)	13,763	16,293	3.763	0.807	10,759	Nov	798	Jan	26,713
6a	SARIMA model (0, 1, 1, 2, 0, 0, 12)	41,842	17,224	11.515	0.785	14,347	Feb	3,577	Jan	30,957
6b	SARIMA model (1, 1, 1, 2, 0, 0, 12)	39,627	16,208	10.904	0.809	11,968	Feb	1,062	Jan	33,432
6c	SARIMA model (0, 1, 1, 2, 1, 2, 12)	41,186	14,262	11.374	0.853	11,621	Sep	638	Dec	24,055
7a	SARIMAX model, 3 exogenous features	39,774	9,320	11.088	0.937	4,678	Mar	322	Dec	15,648
7b	SARIMAX phase 15: 6 exogeneous features	41,235	6,338	11.535	0.971 ✓	2,599	Jul	820	Dec	12,297
8	XGBoost model 5.7	✓ 4,094	✓ 4,858	✓ 1.185	✓ 0.983	✓ 905	Mar	✓ 30	Apr	✓ 7,415
9	LSTM model 5	8,644	10,440	2.436	0.921	5,808	Jun	1,512	Jan	21,081
	Tick marks is	x < = 5%	x < = 5%	x < = 5%	x > = 95%	15%<x< 30%		x < = 5%		x < = 5%

Table 6: performance of the various time series forecasting models

When we compare the best time series forecasting models across all the various models developed in each category the XGBoost is the best of all.

From image 32 it is evident that the error % is less than 4% for the year 2022 with this model. Hence, when we do prediction for the year 2023, we keep a (+/- 5) % range. Within this range the electricity needs to be generated for the year 2023.

date	elec_gen	prediction	error	error_percentage
2022-01-01	377106.442	366997.34375	10109.09825	2.680702
2022-02-01	326930.707	325594.56250	1336.14450	0.408693
2022-03-01	324771.623	325423.90625	652.28325	0.200844
2022-04-01	303323.604	308613.59375	5289.98975	1.744009
2022-05-01	342215.442	331282.68750	10932.75450	3.194699
2022-06-01	380648.538	377431.93750	3216.60050	0.845032
2022-07-01	424013.445	418175.56250	5837.88250	1.376815
2022-08-01	412709.726	411436.84375	1272.88225	0.308421
2022-09-01	350722.260	350043.71875	678.54125	0.193470
2022-10-01	314110.952	316977.46875	2866.51675	0.912581
2022-11-01	322958.838	322681.87500	276.96300	0.085758
2022-12-01	363624.583	358018.62500	5605.95800	1.541688

Image 32: Error % in prediction

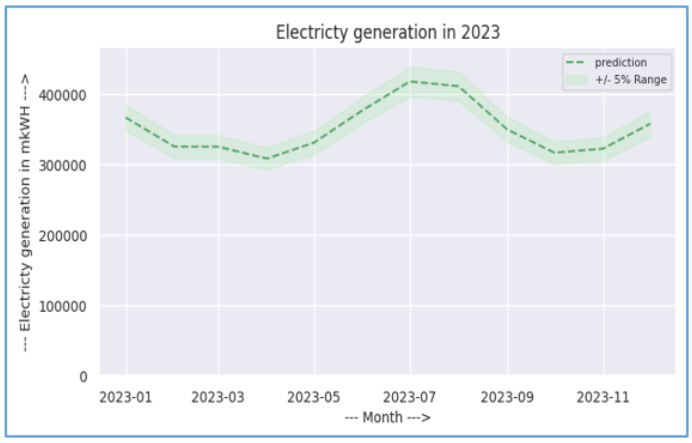


Image 33: Prediction for the year 2023

Future work – we have seen that adding independent features in the model (i.e., multi variate analysis) improved the performance of the model to a greater extent rather than just using tsd as a standalone feature and then using lags / months / seasons (i.e., univariate analysis).

There are other features which can be explored like ‘cost per unit of electricity’ after adjusting inflation etc. There may exist a direct correlation between the ‘cost per unit of electricity’ and electricity consumption and hence electricity generation.
There are other models like VAR - Vector Autoregression which can be explored to predict the electricity generation.

We also tried predicting the electricity generation for a country like India. The challenge was to get good data points. We tried exploring data from various websites like CEIC, GRID-INDIA – National Load Dispatch Center, Indian Ministry of Power to name a few but with little or no success.

13. References

- Machine learning for time series forecasting with python, Francesca Lazzeri, PhD
- A Basic Time Series Forecasting Course with Python, Alain Zemkoho
- Time Series Forecasting Using Python, Kajal Kumari, https://www.analyticsvidhya.com/
- An End-to-End Project on Time Series Analysis and Forecasting with Python, Susan Li, https://towardsdatascience.com/
- Forecasting Univariate Financial Time Series using eXtreme Gradient Boosting, Adrian Axelsson
- The main parameters in XGBoost and their effects on model performance, https://medium.com/
- Grid Search for SARIMAX Parameters, Jessica Forrest-Baldini https://towardsdatascience.com/
- Story telling with data, Cole Nussbaumer Knaflic
- A Guide to Different Evaluation Metrics for Time Series Forecasting Models, Vijaysinh Lendave, https://analyticsindiamag.com/
- Comparison of the different hyperparameter tuning algorithms. Grid search Vs Random search Vs Bayesian optimization By Dr Ernesto Lee
- Monthly electricity consumption and generation in USA for fifty years, US Energy Information Administration
- Monthly USA Real Disposable Personnel Income (RDPI) for fifty years, Jan1973 until Dec2022, FRED economic data
- Keras tuner https://keras.io/keras_tuner/
- LSTM layer https://keras.io/api/layers/recurrent_layers/lstm/
- XGBoost Documentation https://xgboost.readthedocs.io/en/stable/
- HDD, CDD, precipitation, maximum and minimum temperature monthly data for the past fifty years, National Centers for Environmental Information

14. Check list for the final report

Sl No	Check list	Yes / No
1	Is the Cover page in proper format?	Yes
2	Is the Title page in proper format?	Yes
3	Is the Certificate from the Supervisor in proper format? Has it been signed?	Yes
4	Is Abstract included in the Report? Is it properly written?	Yes
5	Does the Table of Contents' page include chapter page numbers?	Yes
6	Is Introduction included in the report? Is it properly written?	Yes
7	Are the Pages numbered properly?	Yes
8	Are the Figures numbered properly?	Yes
9	Are the Tables numbered properly?	Yes
10	Are the Captions for the Figures and Tables proper?	Yes
11	Are the Appendices numbered?	Yes
12	Does the Report have Conclusions/ Recommendations of the work?	Yes
13	Are References/ Bibliography given in the Report?	Yes
14	Have the References been cited in the Report?	Yes
15	Is the citation of References/ Bibliography in proper format?	Yes